1990-12

The implementation of form-based interface for relational database

Partoyo

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/27643

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

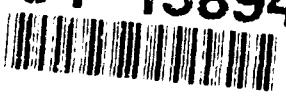
THE IMPLEMENTATION OF FORM-BASED INTERFACE
FOR
RELATIONAL DATABASE

by

Partoyo, Major Indonesian Army

December, 1990

Thesis Advisor: Thomas Wu



SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) 37	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (*If applicable*)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (*City, State, and ZIP Code*)	10 SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11 TITLE (*Include Security Classification*)
The Implementation of Form-Based Interface for Relational Database

12 PERSONAL AUTHOR(S) PARTOYO

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (*year, month, day*)	15 PAGE COUNT
Master's Thesis	From To	1990 December 19	119

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	Form-Based interface, DBMS, ESIS, Form Designer, Pushbutton.

19 ABSTRACT (*continue on reverse if necessary and identify by block number*)

Currently the problem with relational DBMSs is a lack of user-friendly interface. Relational query languages such as SQL and QUEL are not ideal languages for end-users. The forms approach is considered the most natural interface between end-user and database. Several systems based on the forms concept have been designed and implemented. This thesis studies the effectiveness of a form-based visual interface. To evaluate the development process of the form-based applications, this thesis includes the simple implementation of form-based interface using the Form Designer of Superbase-4.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Thomas Wu	22b TELEPHONE (*Include Area code*) (408) 646-3391	22c OFFICE SYMBOL CS/WQ

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

THE IMPLEMENTATION OF FORM-BASED INTERFACE FOR RELATIONAL DATABASE

by

Partoyo
Major, Indonesian Army
B.S., Indonesian Military Academy, 1973

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
[redacted]990

Author: [redacted]
Partoyo

Approved b[redacted]
Thomas Wu, Thesis Advisor

[redacted]
Rachel Griffin, Second Reader

[redacted]
Robert B. McGhee, Chairman
Department of Computer Science

ABSTRACT

Currently the problem with relational DBMSs is a lack of user-friendly interfaces. Relational query languages such as SQL and QUEL are not ideal languages for end-users. The forms approach is considered the most natural interface between end-user and database. Several systems based on the forms concept have been designed and implemented. This thesis studies the effectiveness of a form-based visual interface. To evaluate the development process of the form-based applications, this thesis includes the simple implementation of form-based interface using the Form Designer of Superbase-4.

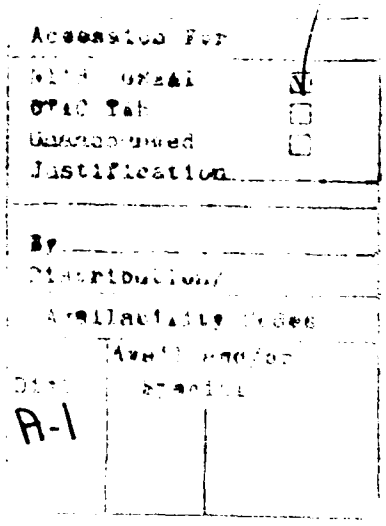


TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis advisor, Professor Thomas Wu and to my second reader LCDR. Rachel Griffin, for their enthusiastic guidance and support. Without many hours of counseling and endless supply of ideas they provided, this thesis could not have been completed.

I would also like to express my thanks to my parents, my wife Parisah, my children Ari, Nia, Novi and Puput, for their understanding, and encouragement, and for the sacrifice they have made during my study in Naval Postgraduate School.

I. INTRODUCTION

A. MOTIVATION

Database technology is having a major impact on the use of computers and is playing a critical role in the development of information systems today. There are several reasons for this.

First, a database can store large volumes of corporate operational data. Second, a database can be queried on an ad hoc basis, making it the foundation for decision-support systems(DSS). Data stored in a database can be readily accessed and processed. Third, a database can be implemented on computers of all sizes, making it feasible for almost any business or organization.

A database can be generated and maintained either by a group of applications programs or by a database management system (DBMS). The main criteria used to classify a DBMS is the data model on which the DBMS is based. Those data models are: 1) relational, 2) network and 3) hierarchical.

In recent years, relational systems have become the prevalent database management systems. They are based on a solid mathematical foundation and provide better query languages than do network and hierarchical systems.

Although relational query languages such as SQL and QUEL are better languages than those for network and hierarchical systems, they are still not ideal languages for end-users. Desirable characteristics for a database development tool include reusability, modifiability, and extensibility. The development language should be powerful enough to encode complex

ideas and relationships with small amounts of code. Superbase-4 is a language that meets these characteristics.

Superbase-4 is a comprehensive, general purpose relational database management system designed to operate in the Microsoft Windows graphical environment [Ref. 7:p. 1-1]. As a Windows application, Superbase-4 may be executed simultaneously with Windows graphical and business applications, and can exchange data with them. Superbase-4 can be driven either by mouse or by keyboard.

The primary goal of this thesis is to study the effectiveness of a form-based visual interface to databases using the Form Designer of Superbase-4. Form Designer is the element of Superbase-4 that provides facilities for creating a form. A form is the user's application view of data. To achieve the goal we include a simple implementation of an Educational Scheduling Information System (ESIS) using Superbase-4. This system produces information about faculty members and the courses that are assigned to them. Underlying the system is a relational database with 5 relations: FACULTY, COURSE, DEPARTMENT, PUBLICATIONS and OFFERING.

B. THESIS OVERVIEW

Chapter II will present some background information for the thesis, beginning with a discussion of relational databases. Further discussions will address relational algebra and the problems of the relational approach. Next we will provide an overview of the Microsoft Windows Operating Environment, including Window's Dynamic Data Exchange (DDE) which allows applications to communicate and exchange information with other Window's applications. Lastly, we will describe Superbase-4 which operates under the Microsoft Windows graphical environment.

Chapter III explains the design goals of implementation, the Educational Scheduling Database structure and the development of form-based user interfaces.

Chapter IV evaluates the application development process that uses a form-based approach. The chapter addresses form-oriented database design, the relational category of Superbase-4 and the evaluation of the Form Designer. Chapter V will present conclusions.

II. BACKGROUND

A. RELATIONAL DATABASE

As stated in Chapter I, the majority of present-day databases are based on the relational approach. The relational model was first proposed by Dr. E. F. Codd (1970). Due to the rapid improvement of database technology, users with little computer experience and even less database experience must frequently interact with an organizational database.

One problem with current systems is a lack of user-friendly interfaces. Relational query languages such as SQL and QUEL are not ideal languages for end-users. This section will show some problems of SQL. We will demonstrate the need for a visual database interface with query facilities for accessing databases that are easy to learn and use.

1. Relational Data Structure

To explain the relational data structure, we will provide a simple example of a relational database. Figure 2.1 shows a relation of data that is organized into three tables: FACULTY, COURSE, and OFFERING. The FACULTY table contains a faculty identification, a faculty last name, a faculty first name, a faculty SSN and a faculty phone number. The COURSE table contains a course code, a course name and the credit hours for the course. The OFFERING table contains a course code, a section number, a quarter and year when the course is offered and faculty identification of the instructor.

F_ID	L_NAME	F_NAME	SSN	PHONE
F01	Smith	John	123456789	123-4567
F02	Shelly	Paulin	222333444	321-1234
F03	Correl	William	999888777	333-444

(a) FACULTY Relation

F_ID	C_NAME	CREDIT
CS2970	ADA Programming	4.0
CS3320	Intro to Database	3.5
CS3502	Computer Network	4.0
CS3310	A.I	4.0

(b) COURSE Relation

C_ID	S_NUM	QUARTER	YEAR	INSTR
CS2970	1	Summer	90	F01
CS2970	2	Summer	90	F02
CS3505	1	Spring	89	F03
CS3320	1	Summer	90	F02
CS3320	2	Summer	90	F01
CS3310	1	Winter	89	F03

(c) OFFERING Relation

Figure 2.1 FACULTY, COURSE and OFFERING Relations

a. Terminology

The relational database model is based on the concept that data is organized and stored in a two dimensional table called a relation. Each row in the table represents a record and is called a tuple. Each column represents a field and is called an attribute. The entire table is roughly equivalent to a file. Certain restrictions are imposed on relations. First, attributes are single-valued; neither repeating groups nor arrays are allowed. Second, entries in any column are all of the same kind. Third, each attribute has a unique name, and attribute positions are insignificant. Finally, no two tuples in a relation may be identical.

b. Keys of Relation

The key is the attribute or set of attributes that uniquely identifies tuples in a relation [Ref. 2:p 139]. From Figure 2.1 the F_ID is the key of the FACULTY relation. Its unique values distinguish each tuple within the relation.

A key may be the composition of more than one attribute. Such a key is referred to as a composite key. For example, in the relation OFFERING in Figure 2.1, the combination of the attributes C_CODE, S_NUM, QUARTER and YEAR serves as the composite key.

It is possible for a relation to have more than one key. In such a case we would say that the relation has multiple candidate keys. We would choose one of those candidate keys to be the primary key, and the others would be alternate keys. In the FACULTY relation F_ID and SSN are candidate keys. F_ID is the primary key and SSN is the alternate key.

When an attribute in one relation is a primary key for another relation, the attribute is called a foreign key. For example in Figure 2.1 within OFFERING relation,

attribute INSTR is a foreign key, because attribute INSTR is the primary key of the FACULTY relation. A foreign key value represents a reference to the tuple containing the matching primary key value [Ref. 3:p. 282].

2. Relational Algebra

The relational algebra is a collection of operations that are used to manipulate relations [Ref. 1:p. 148]. These operations are used to select tuples from one or more relations to achieve a desired result. The result of each operation is a new relation, which can be further manipulated by the relational algebra operations. Relational algebra is hard to use because it is procedural. That is, when using relational algebra we must know not only what we want, but also how to get it [Ref. 2:p. 306].

The relational algebra operations are usually divided into two groups of operators: traditional operators and special relational operators.

a. Traditional Operators

(1) Union. The union of two relations is formed by combining two tuples forming one relation with those of a second relation to produce a third relation. Duplicate tuples are eliminated. For this operation to make sense, the relations must be union compatible. This means that each relation must have the same number of attributes and that the attributes in corresponding columns must come from the same domain. For example, if relation A is the set of faculty members from the computer science department, and relation B is the set of faculty members who teach CS2970, then A UNION B would be relation C, the set of faculty members from computer science department or faculty members who teach CS2970 or both.

(2) Difference. When relation C contains tuples that occur in relation A but not in relation B, and the tuples that occur in relation B but not in relation A, then relation C is said to be the difference of relation A and B. Using the example in (1) above, then relation C will contain the faculty members from the department of computer science who are not teaching CS2970.

(3) Intersection. If relation C contains tuples that are in relations A and B, then relation C is called the intersection of relation A and relation B. The intersection of relations A and B will contain the faculty members who teach CS2970.

(4) Cartesian Product. Relation C is said to be the cartesian product of relation A and relation B if every tuple in C is a concatenation of each tuple in relation A with every tuple in relation B. Suppose relational A has m tuples and B has n tuples, then relational C will have m x n tuples.

b. Special Relational Operators

(1) Projection. Projection is an operation that selects specified attributes from a relation [Ref. 2:p. 313]. The result of the projection is a new relation that has the selected attributes. Thus projection takes a vertical subset (columns) of a relation. Any duplicate tuples within the attributes selected are eliminated. Projection can also be used to change the order of attributes in a relation. Figure 2.2 shows the projection of the FACULTY relation on the last name and phone number attributes.

L_NAME	PHONE
Smith	123-4567
Shelly	321-7654
Correl	333-4444

Figure 2.2 Projection of FACULTY Relation

(2) Selection. The selection operator take a horizontal subset (row) of a given relation. A tuple may be extracted from the relation by specifying the relation name followed by keyword WHERE, followed by a condition involving attributes. Figure 2.3 shows the selection of relation FACULTY WHERE L_name = "Smith".

F_ID	L_NAME	F_NAME	SSN	PHONE
F01	Smith	John	123456789	123-4567

Figure 2.3 Selection of FACULTY Relation

(3) Join. The join operation is a combination of the product, selection and projection operations. The join between two relations A and B, operates in three steps. First, take the product of A times B, then do a selection to eliminate tuples that do not meet the criteria of selection. Finally remove duplicate attributes with projection. This is called a natural join. If duplicate attributes are not removed it is called an equi-join. The join operation is important for any relational database with more than a single relation because it allows the combination of related tuples from two relations into a single tuple. Figure 2.4 shows the natural join of the FACULTY and OFFERING relations.

F_ID	L_NAME	F_NAME	...	C_ID	S_NUM	QRTR	YR
F01	Smith	John	...	CS2970	1	Summer	90
F01	Smith	John	...	CS3320	2	Summer	90
F02	Shelly	Paulin	...	CS2970	2	Summer	90
F02	Shelly	Paulin	...	CS3320	1	Winter	90
F03	Correl	Paulin	...	CS3505	1	Spring	89
F03	Correl	Paulin	...	CS3310	2	Winter	89

Figure 2.4 Natural Join of FACULTY and OFFERING Relations

3. SQL A Relational Database Language

Originally, SQL was called SEQUEL (Structured English Query Language) and was designed and implemented at IBM as the interface for an experimental relational database system called SYSTEM R.[Ref. 1.:p. 175]. SQL is a non-procedural language or declarative language. In such a language, we construct a query by specifying what data is to be retrieved rather than specifying how to retrieve the data. The terms, table,row, and column are used to represent relation, tuple and attribute, respectively. SQL can be used interactively as a query language or it may be embedded in application programs.

a. SQL Projections

To form a projection with SQL, we list the columns we want to see in a specific format. For example, the projection FACULTY(F_ID,L_NAME,PHONE) is created by specifying:

```
SELECT   F_ID,L_NAME,PHONE
FROM     FACULTY
```

b. SQL Selections

The relational algebra selection operator is specified in SQL form as SELECT-FROM-WHERE structure, and this is the fundamental structure of SQL statements. Suppose we want to retrieve all instructors who teach CS2970, then the query would be:

```
SELECT   INSTR
FROM     OFFERING
WHERE    C_ID = "CS2970"
```

c. Joining with SQL

To produce the name of instructors who teach CS2970, we need to join the FACULTY table and the OFFERING table. It can be done by the following statements:

```
SELECT   FACULTY.L_NAME,FACULTY.PHONE
FROM     FACULTY,OFFERING
WHERE    FACULTY.F_ID = OFFERING.INSTR AND
         OFFERING.C_ID = "CS2970"
```

d. Nested Queries

Some queries require that existing values in the database be fetched and then used in a comparison condition. For example, we want to find instructors who teach the course with credit hours more than 3.0. The query would be:

```
SELECT   L_NAME,PHONE
FROM     FACULTY
WHERE    FACULTY.F_ID IN (
             SELECT      INSTR
             FROM        OFFERING,COURSE
             WHERE       OFFERING.C_ID = COURSE.C_ID
                         AND
                         COURSE.CREDIT > 3.0)
```

The first nested query selects the faculty ID's of those who teach the course with credit hours more than 3.0. In the outer query, we select a FACULTY tuple if F_ID value of that tuple is in the result of the first nested query. We also may write this query using the CONTAINS operator as follows:

```
SELECT   L_NAME,PHONE
FROM     FACULTY
WHERE    ((SELECT     C_ID
           FROM       OFFERING
           WHERE      F_ID = INSTR)
           CONTAINS(
                SELECT      C_ID
```

```
        FROM    COURSE
        WHERE   CREDIT > 3.0))
```

Another way to write a nested query is by using the EXISTS function. For example we would like to list all faculty members who are not teaching any courses in summer 1990.

```
SELECT  L_NAME
FROM    FACULTY
WHERE   NOT EXISTS(
            SELECT      *
            FROM        OFFERING
            WHERE       F_ID = INSTR AND
                        QUARTER = "SUMMER" AND
                        YEAR = "90")
```

The correlated nested query retrieves all OFFERING tuples for summer 90 and matches them with the current tuple of the FACULTY relation. If none exists then the current FACULTY tuple is selected and L_NAME is retrieved.

4. The Problems of SQL

In light of the previous examples we can see that SQL is not easy to use. This is especially true for non-experienced users. The reasons are:

a. SQL has no easy-to-use facility for saving and reusing temporary results of queries. It forces one to nest sub-queries to accomplish passing of results. Nested queries are very complicated structures as we saw in the previous examples. They are not easy to use even for experienced users.

b. In many instances the users must specify how to get the result rather than stating what the desired result is. This is in contradiction with the philosophy of SQL as a non procedural based language.

c. SQL does not address the notion of foreign keys and the related notion of referential integrity. It is possible to leave a database in an inconsistent state by using foreign key values that do not obey the referential constraint. Referential constraint means that values of a given foreign key must match values of the corresponding primary key [Ref. 3:p. 282]. For example, from Figure 2.1 we know that INSTR is foreign key in the OFFERING relation, because INSTR is the primary key of the FACULTY relation. Suppose we update the faculty ID of John Smith from F01 to F04 in the FACULTY relation. SQL will not automatically update the value of INSTR for John Smith in the OFFERING relation. Now the FACULTY relation and the OFFERING relation are in an inconsistent state.

d. SQL does not show the meaningful relationships among tables to the user. Furthermore, joins that semantically make no sense are still allowed and lead to incorrect results.

B. MICROSOFT WINDOWS GRAPHICAL ENVIRONMENT

Windowing interfaces were started by the XEROX Palo Alto Research Center (PARC) in the mid-1970's. They were popularized by Apple Computer, and later by Microsoft and other developers. The Windows environment offers numerous advantages such as hardware independence and dynamic data exchange as well as a user-friendly interface and an intuitive graphical interface.

1. Graphical User Interface

The Windows interface is based on graphical objects such as menus, dialog boxes and icons. In the Windows environment menus and icons have replaced the cryptic DOS command line syntax. User may interact with their programs by using a mouse or other pointing device instead of a keyboard. Figure 2.5 shows the typical Window format.

Windows also provides the Graphics Device Interface (GDI), which contains routines for drawing menus, windows, dialog-boxes and many other graphic objects.

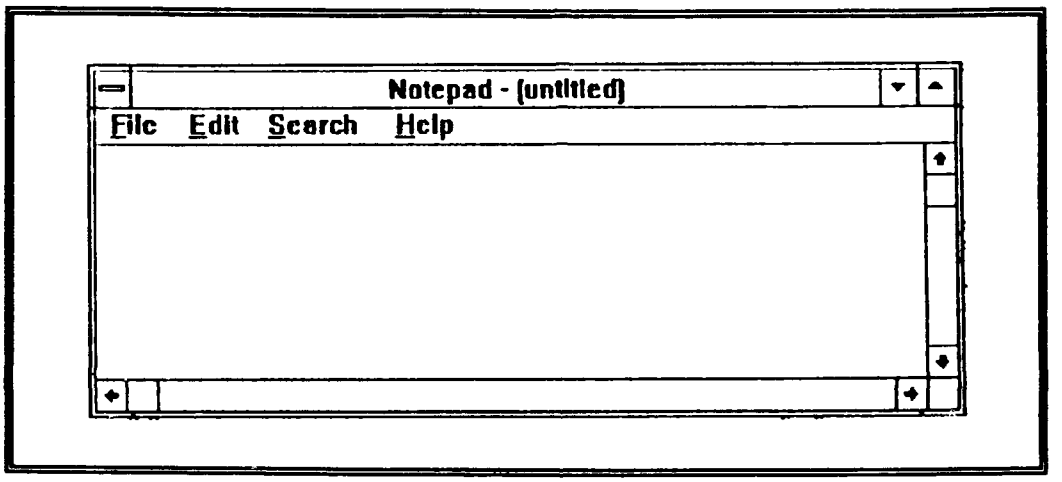


Figure 2.5 Typical Window

2. Hardware Independence

The Windows environment is hardware independent.This means that if our program can be run under Windows on a particular computer, then it will also run on any computer that Windows supports. Screen layouts will work on any monitor, and printer output may be obtained on any brand of printer. This capability will relieve the programmer of the burden of writing code to support each peripheral device available.

3. Dynamic Data Exchange (DDE)

In the Microsoft Windows environment, different applications may be linked together so data can be easily transferred between them automatically. Information can be passed without the user explicitly asking for it.

Programs using DDE must follow the protocol established by Windows to pass the information. Normally, one application does the asking, and another responds. The asking application is often referred to as the client program, and the responding application is called the server.

Many applications require a more sustained relationship between programs, in which both data and commands are passed repetitively and for longer periods. This kind of DDE relationship is called a conversation.

C. SUPERBASE-4

One of the database managers that operates under the Microsoft Windows graphical environment is Superbase-4. As a Windows application, Superbase-4 may be executed simultaneously with other Windows applications and can exchange data with them.

Superbase-4 includes facilities for the interactive development of form applications using its Form Designer and provides an interface to the relational database system. A form is the user's application view of data which consist of a background of graphic objects such as areas or lines, onto which fields from one or more existing database file may be placed [Ref. 8:p. 1-1]. We may also include calculation formulas and commands on a form, allowing us to construct a complete application with no programming.

1. Getting Started with Superbase-4

To start working with Superbase-4 we must load Microsoft Windows. Then the Superbase-4 program file SB4W.EXE may be run. When this program is executed, Superbase-4 presents a Work Area with pull-down menus at the top and the browsing control at the bottom. (See Figure 2.6).

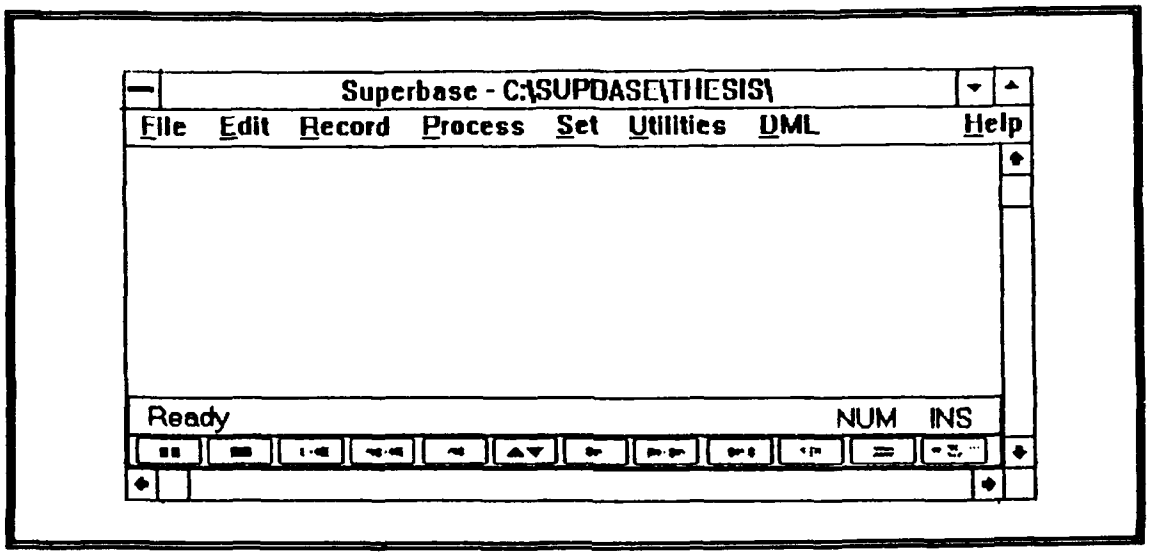


Figure 2.6 Superbase-4 Work Area

a. The Superbase-4 Work Area

Superbase-4 uses the Work Area to show data, either with or without a form. With a form, data from several files can be shown at once. Without a form, Superbase-4 shows just the records in the current file, as well as the results of some other operations.

b. The Superbase-4 Menus

Superbase-4 provides 7 menus: File, Edit, Record, Process, Set, Utilities and DML. For each menu there are some submenus (Figure 2.7).

Almost all operations can be accessed from pull-down menus that allow one to create a new file, modify an existing file, create a new index for an existing file, create a query, write a new program, execute an existing program and perform other tasks.

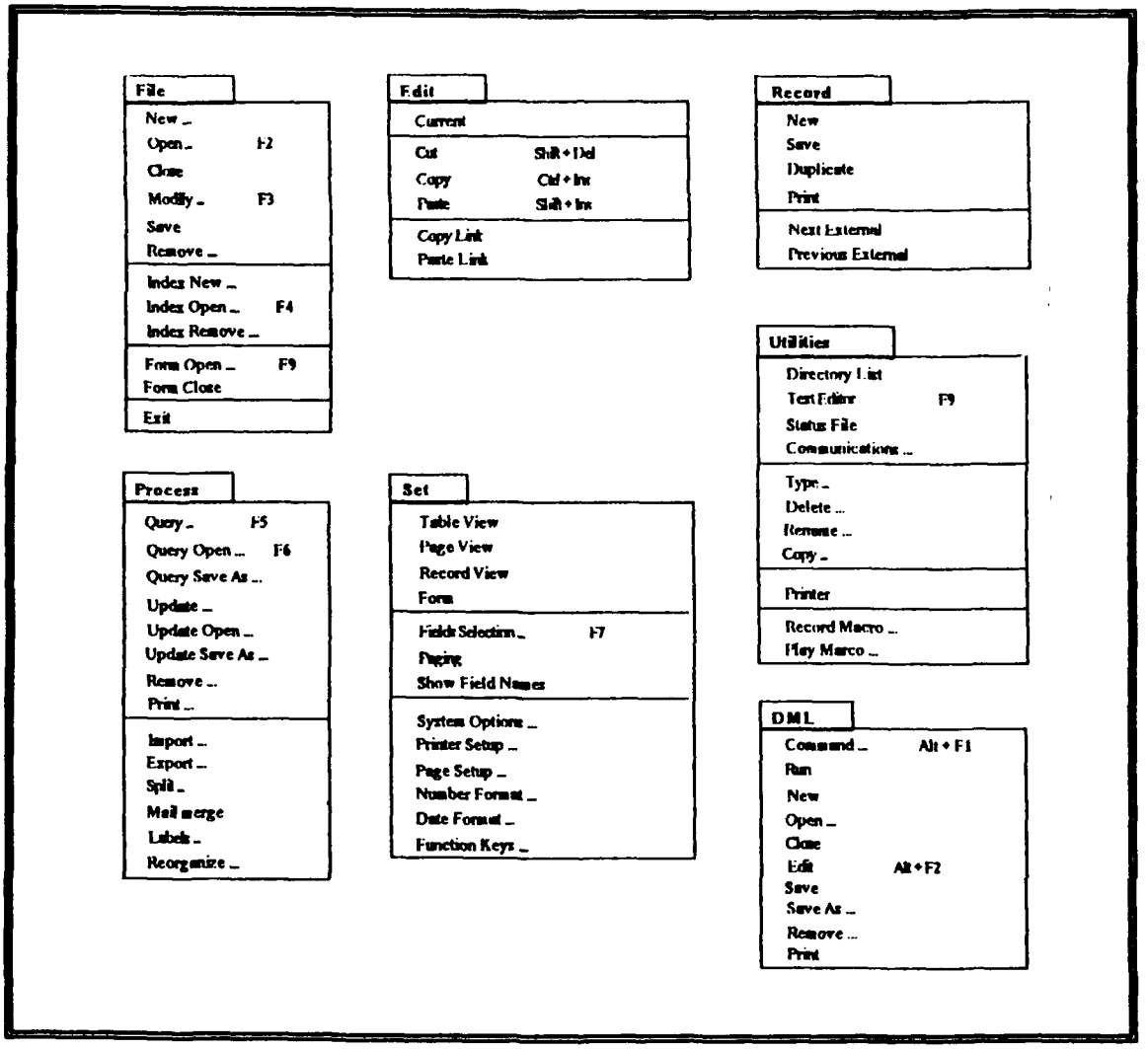


Figure 2.7 The Superbase-4 Menus

c. The Browsing Controls

The browsing controls provide a set of tools for instant and intuitive file management. With the browsing controls users may select next record, previous record, first record, last record, rewind file, fast forward, pause or stop by clicking a mouse or pressing the appropriate key. This feature is easy to use for non-experienced users. Figure 2.8 shows the browsing controls.

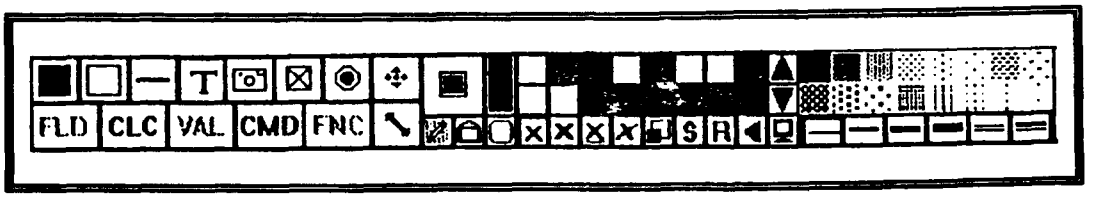


Figure 2.8 The Browsing Controls

2. Form Designer

The Form Designer is a design system for producing input and browsing screens, forms for printing, and reports. [Ref. 8 p: 1-2]. The Form Designer program is called SBFD4W.EXE. It can be executed from the Windows Desktop or from the command line like any other Windows applications. When the program is executed, Superbase-4 presents the Form Designer menus.

a. The Form Designer Menus

The Form Designer menus contain all commands for controlling the system. There are 5 menus: File, Edit, Page, Define and Report Generator, and each menu has sub menus (Figure 2.9). Performing a special operation from this menu is achieved in the same way as in the Superbase-4 Menus.

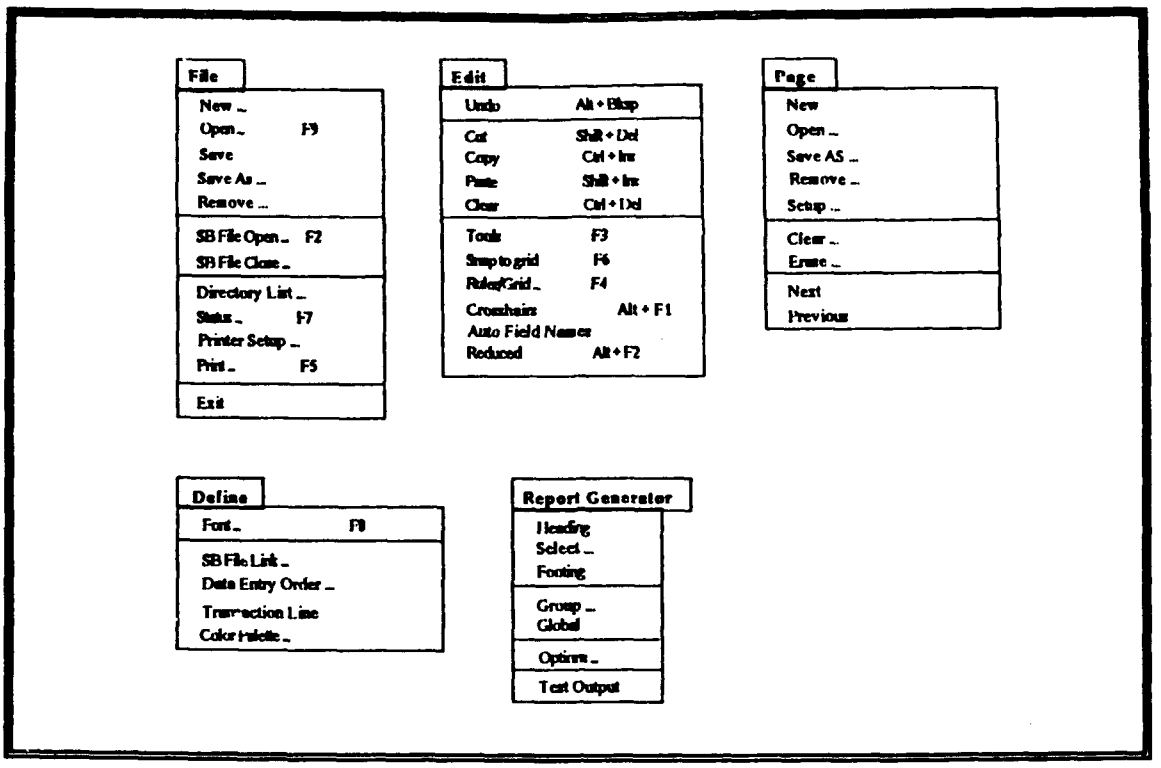


Figure 2.9 The Form Designer Menus

b. The Toolbox

The Form Designer toolbox is the most important resource for the application designer. The Form Designer toolbox provides easy and continuous access to the object types and editing choices needed by the application designer. Objects can be fields of opened files, text, images, lines, boxes or areas. Figure 2.10 shows the Form Designer toolbox.

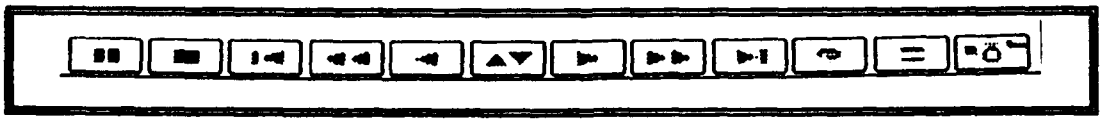


Figure 2.10 The Form Designer Toolbox

The left-hand part of the toolbox shows the graphical, data and logical object types. Next to them are the tools for moving and sizing objects. In the center of the toolbox are the color selection tools and groups of tools for selecting pen and paper, rounded corners and borders. Next to these are tools for display only, text style, read only, currency calculation type, image scaling, and field justification.

The right hand part of the toolbox includes palettes for color, area pattern, and line pattern.

c. Placing Objects on The Form

The order in which types of objects are placed on a form is largely a matter of individual preference. The user simply places the object at the location desired using a mouse. Once an object is on the form, it may easily be sized or moved using the Form Designer toolbox.

By placing a field on a form we may enter data into the database as well as retrieve it. Fields can be selected from any existing database file, and positioned anywhere on the form. When a form consists of fields from more than one file we have to join the files using the Define SB Link command. Each pair of files to be linked must have a pair of indexed fields that contain matching data.

A field may be designated as read only to protect it from being changed, or display only to stop it from being printed. When fields will not fit on a one-page form, then it is possible to build a multi-page form by using the Page menu. The Page menu includes commands for creating a new page, opening and saving pages, clearing and erasing pages, and for switching between pages.

d. Transaction Lines

Some applications include one-to-many relationships between records in different files. In a form, we can define a group of fields as the "many" part of such a relationship, using the Transaction Line command. Typically, each transaction is a row of fields from a single record belonging to the file at the "many" part of a one-to-many relationship.

We can define the number of rows and the number of columns that will appear on the form by inserting the number in the transaction lines box of the Transaction Lines Dialog.

For a transaction to work, a link between a field in the master file and a field in the transaction line must be established using the Link command on the Set menu.

e. Commands and Controls

The Form Designer includes facilities for adding commands, pushbuttons, check boxes, and radio buttons to forms.

(1) A command is one or more DML statements entered as a single line. Commands are executed when the insertion point passes through them. A command may be used for validation, immediately after the field to which it relates is entered, or to load a DML program.

(2) A pushbutton is a graphic object that executes a user-specified command when the user clicks it. Visually a pushbutton is a rectangle with rounded corners, containing a label that indicates its purpose. Pushbuttons may be used for many purposes, such as to trigger complex processing or to switch to another form.

(3) The other kinds of form control, radio buttons and check boxes, are attached to fields or variable calculations. Their purposes vary.

3. Query and Report

a. Queries

The Query command is a powerful and versatile tool for data selection. It has five main areas of application:

(1) Creating complex filters. A filter is a set of conditions for the record to be selected. It is very common for applications to retrieve the same set of records over and over again using the same or similar criteria. For this kind of application, Query is the best tool, because we can create a complex filter for the query, save the query on the disk and execute it whenever needed.

(2) Multi-file applications. By setting up a relational link in the Query Filter, we can select several files at the same time.

(3) Reporting. The report features available in Query provide additional information about the results of a search, such as record counts or field totals.

(4) Sorting. Query output can be sorted into any order, either ascending or descending. We can also specify several levels of sorting, using a different field for each level.

(5) Output redirection. The output from a query can be directed to one of four possible destinations: the screen, the printer, a text file, or a new database file.

The Query command is activated by selecting the Query Open command from the Process menu. After selecting the desired query from the list, Superbase-4 presents the Query Definition Dialog as illustrated in Figure 2.11.

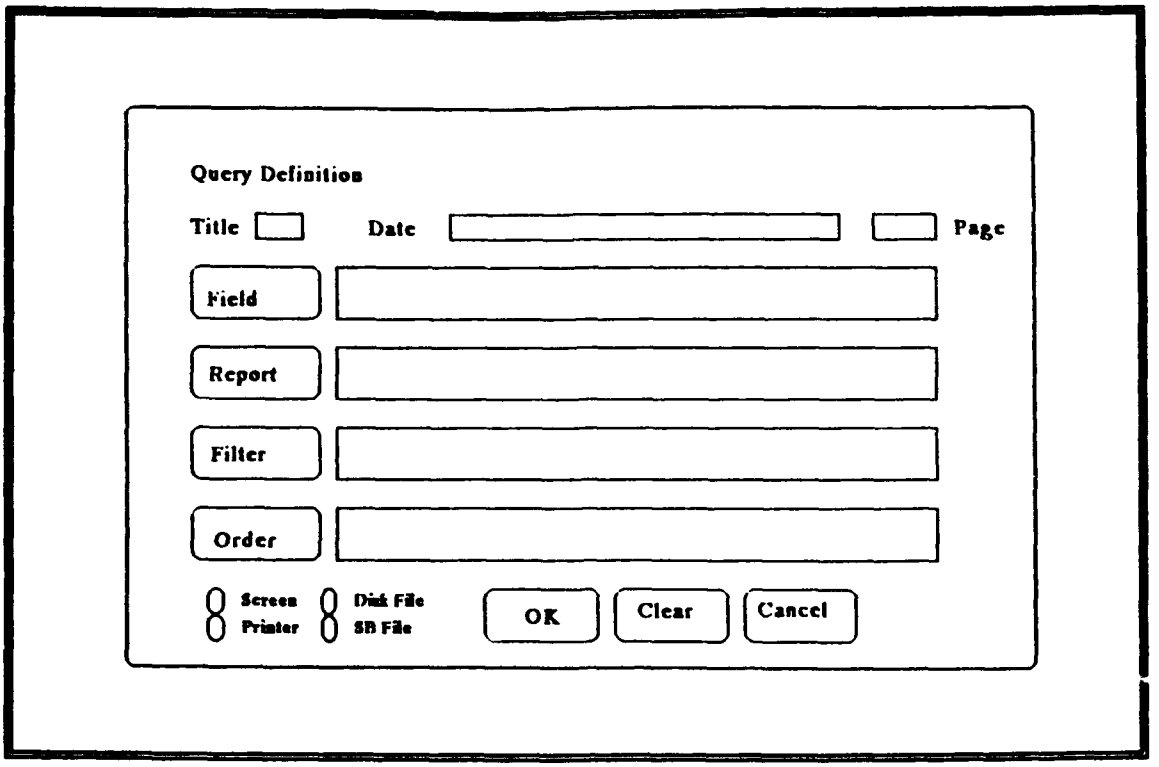


Figure 2.11 The Query Definition Dialog

The Query Definition dialog has three parts, at the top is the Title Definition, below is the Query Command Panel, and at the bottom left are the selected destinations for the output from the query.

The four lines of the Query Command Panel do most of the work of the Query command. Each line defines one of the elements of a query: output fields, report features, filter conditions and sorting order. To run the query click OK button or press ENTER key.

b. Reports

In Superbase-4, reporting is considered to be an extension of the program's query facilities. Explanatory text, such as headings and footings, is combined with functions such as subtotaling and mean calculation to produce more informative output (Figure 2.12).

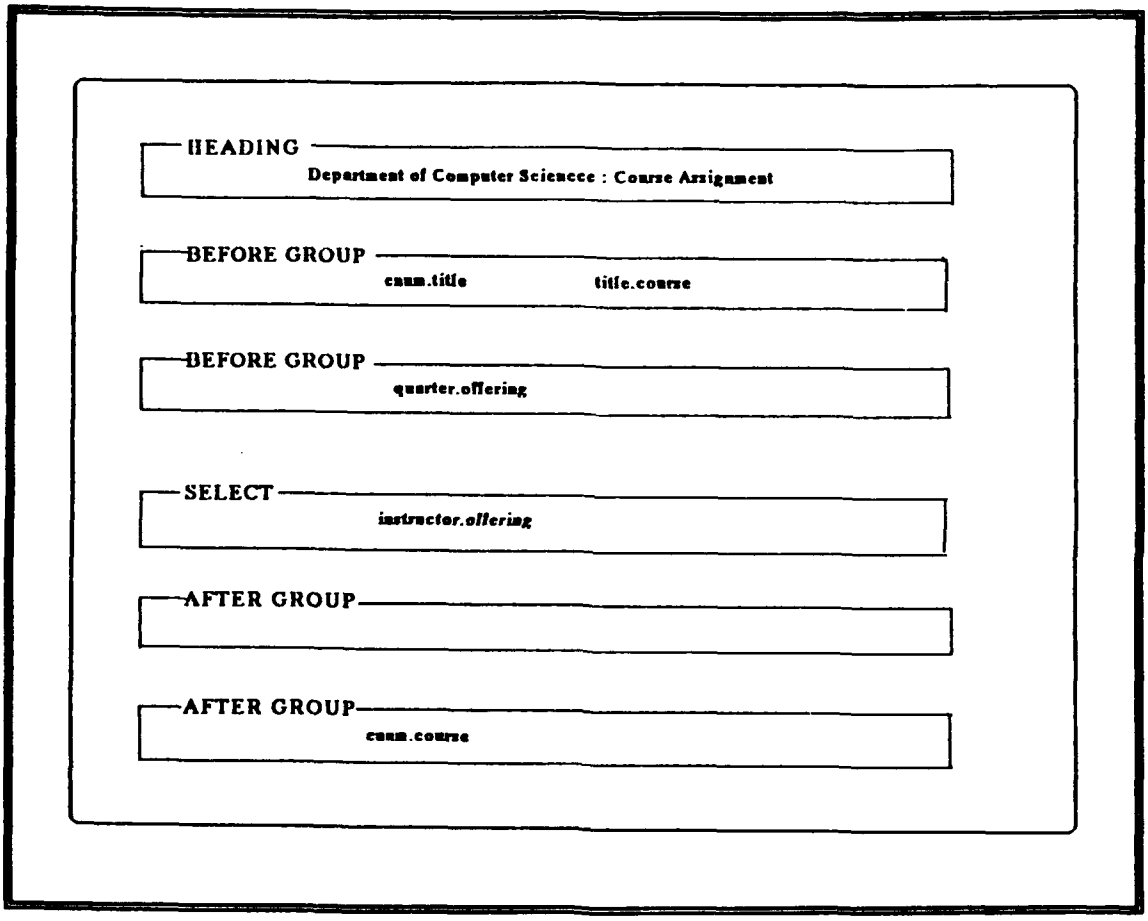


Figure 2.12 The Example Report Form

The Form Designer provides a set of commands for generating reports. When a report is saved, the Form Designer generates a DML program and stores it on disk. This program may be edited from the DML command menu or from the Form Designer.

4. Data Management Language (DML)

DML is a facility for technical users to accomplish complex queries such as one-to-many forms browsing or multi-column label printing. DML is based on the programming language Basic. It includes most of the standard Basic commands and functions, but supplements them with a large number of commands and functions that are specific to database management. There are two modes of operation: direct mode and program mode.

In direct mode, DML executes instructions as soon as we type them. To choose direct mode, select the Command option on the File menu.

In program mode, DML does not execute commands as we enter them. Instead commands are stored in memory and executed only when the program is run. Superbase-4's Program Editor provides facilities for the development of programs in DML code. A program line can be up to 255 characters.

a. Variables

There are three types of variables in DML: string variables, numeric variables, and arrays. String variables are used to hold ASCII characters. The maximum length of data that can be held in a string variable is 4000 characters. A string variable name must end with the "$" character.

DML's numeric variables hold numbers to 14-figure accuracy. If displayed or printed, they are shown in the current default numeric format. The default may be changed. Numeric variables name must end with "%" character.

DML supports string and numeric arrays with up to three dimensions. The maximum number of elements in an array is limited only by the amount of memory available.

b. Functions

Functions form one of the largest groups of DML keywords. Most of them perform a calculation on a number or a string, but there are also functions that give information about some aspects of the system. RECCOUNT for example, returns the number of records in a file, FREE returns the amount of free memory space.

c. Operators

DML provides three types of operators: arithmetic, relational and string. The arithmetic operators are addition, subtraction, multiplication, division, exponentiation and modulo arithmetic. Parentheses can be used to change the order in which the different parts of an expression are calculated.

Relational operators compare the values of two numbers or two strings, and return a result which is either true or false. The relational operators are

>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
<>	not equal to
=	equal to
LIKE	pattern matching operator for string
CONTAINS	pattern matching operator for text file

d. Running a Program

To run the program already loaded in memory, select Run from the File or DML menu. If there is no program in memory, Superbase-4 displays the program file open dialog. We can select the required program and load it into memory for execution.

Programs can also be executed automatically when we run Superbase-4 by specifying a "start-up" program. To create a start-up program, first we must store the program in a directory where Superbase-4 will find it: either the start directory or the directory which is current when Superbase-4 is loaded. Second we must save the program under the file name START.SBP.

III. IMPLEMENTATION

A. THE GOAL OF IMPLEMENTATION

In Chapter II we discussed several facilities of Superbase-4 including the Form Designer. Using the Form Designer we may develop the form-based application interactively with no programming. To study the effectiveness of developing the form-based interface, we present the experience of developing a simple form-based application using Superbase-4.

The goal of the implementation is to show how easy it is to use Superbase-4 to develop a form-based application. We discuss how the Form Designer, DML and other facilities have helped us to improve the quality and reduce the quantity of our code.

B. THE EDUCATIONAL SCHEDULING INFORMATION SYSTEM (ESIS)

1. The Purpose of the System

The purpose of the Educational Scheduling Information System is to automate the scheduling of courses. This system illustrates how a Superbase application can be developed with a minimum of programming.

2. The Database Structure

We first identify three objects which are needed by the system, Faculty, Course and Department. Figure 3.1 illustrates the relationship between these objects.

a. Faculty

This object consists of data such as faculty ID, last name, first name, phone, teaching preference, department code, biography and publication. Publication is a multi-value attribute. Thus, the Faculty object will be represented by two relations. One will

contain general information about a faculty member. The other will contain the publications associated with the faculty member. The relations have a many-to-many relationship and total/partial participation. This means that a publication must be associated with a faculty member but a faculty member is not required to have any publications. The format of the two relations are

FACULTY(Fac_ID, Lastname, Firstname, Phone, Teaching_pref, D_code, Biography).

PUBLICATION(Fac_ID, Publication, Author, Title, Volume, Number, Year, Number_pages).

b. Course

The Course object consists of data such as course number, course title, course credits, syllabus and coordinator of the course. The relationship between Course and Faculty is called Offering. Its attributes are: section number, year, quarter, constraints, number of students who attend the course at the beginning and at the end of enrollment and SOF. The relation between Course and Faculty is many-to-many and has total/partial participation. This means each course must be taught by at least one faculty member and each faculty member may or may not teach any of the courses. The relation formats are

COURSE(C_num, Title, Units, Coordinator, Prerequisites, Syllabus)

OFFERING(C_num, Section, Quarter, Year, Instructor, Constraints, Beg_cnt, End_cnt, SOF).

c. Department

The last object of this system is Department. It consists of data such as department code, department name, chairman and phone number. Two relationships exist between Department and Faculty, namely WORK_ON and CHAIRMAN.

The WORK_ON relationship is a one-to-many relationship and has total/total participation. This means that each department must have at least one faculty member who works in that department, and each faculty member must work only in one department.

The CHAIRMAN relationship is a one-to-one relationship and has total/partial participation. This means that each department must have one and only one faculty assigned as chairman and not all faculty members must be chairmen.

The CHAIRMAN relationship is represented by placing a CHAIRMAN attribute in the Department relation. The WORK_ON relationship is established by placing the D_CODE attribute in the Faculty relation. The relation format for the Department relation is

DEPARTMENT(D_code,D_name, Chairman, Phone).

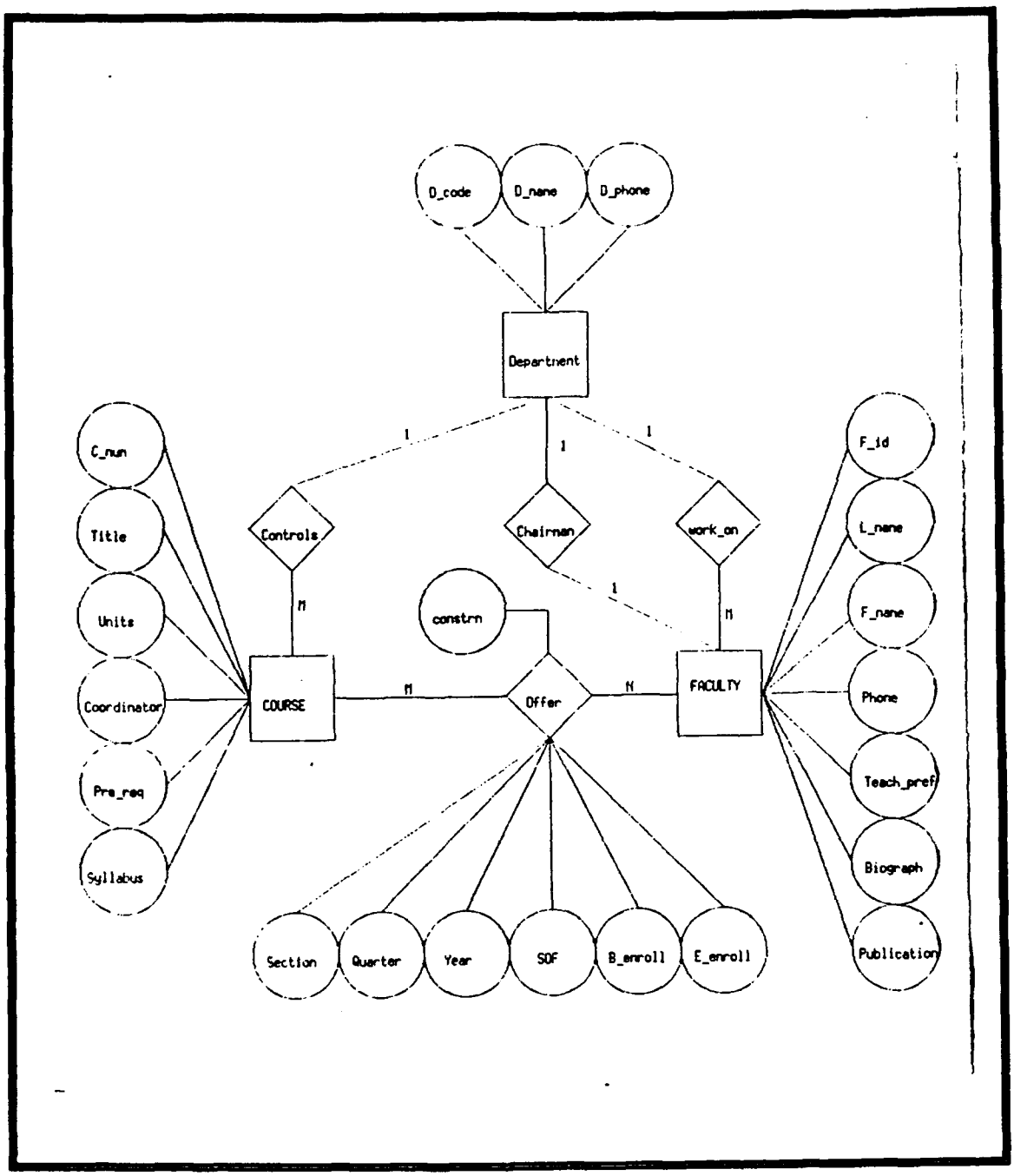


Figure 3.1 The Entity Relation Diagram

3. The Menu Structure

Menu selections are attractive because they can eliminate training and memorization of complex command sequences [Ref. 9:p. 86]. When the menu items are written using familiar terminology, users can select an item easily.

For our proposed system, we organize the menu selections in a hierarchical structure. [See Figure 3.2]. The Main menu is located at the top level. There are seven option menus that can be selected from the Main menu: Faculty, Course, Offering, Department, Update, Report and Others.

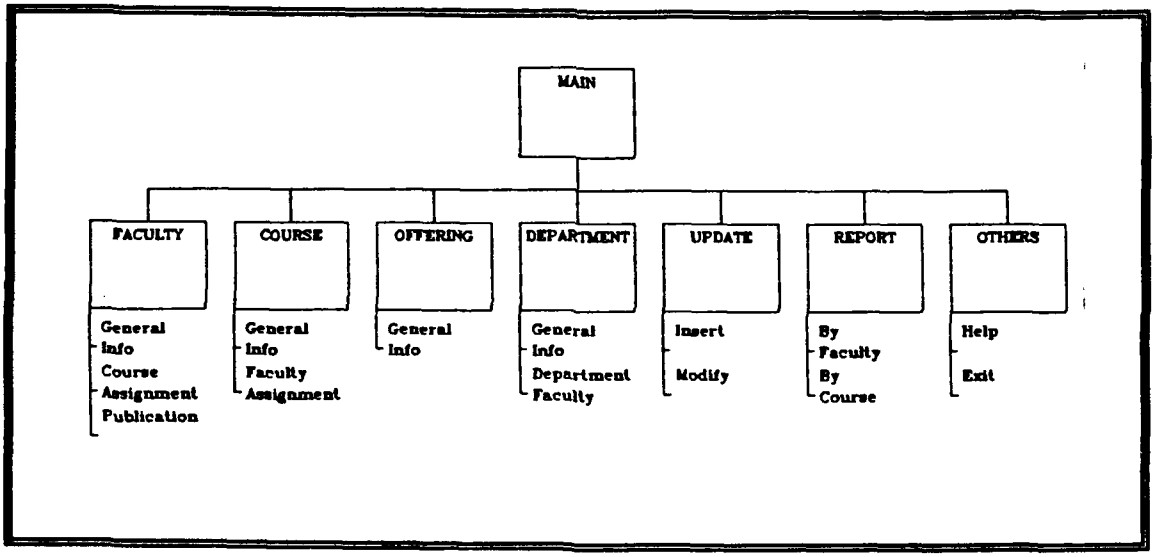


Figure 3.2 The Menu Structure

The Faculty menu provides three options: display the general information of the faculty member, display the course assignment and display the publications of the faculty member. The Course menu provides two options: display the general information of the course and display the faculty assignment. The Offering menu consists of one option, which is display

the general information of the offering relation. The Department menu provides two options: display the general information of department and display the department of a given faculty member. The Update menu consists of two submenus: insert a new record and modify an existing record. There are five options that can be selected for each submenu according to the file to be updated. The Report menu contains two options: report by faculty member and report by course. The Others menu contains two options: help and exit.

4. Defining The Files

From the previous discussion we know that to develop the proposed system we need five relations, namely Faculty, Publication, Course, Offering and Department. Now, let us discuss how to create the files of these relations.

There are several steps required to set up a new file using Superbase-4. First, from the Superbase-4 Work Area we select New under the File menu and type in a new file name without any extension. Superbase-4 automatically adds SBF extension when we save the file. FACULTY, PUBLIC, COURSE, OFFER and DEPART are the names of five files needed by the system. For these files we may add passwords for protection, but to keep our application simple, we do not add any passwords.

Second, we define the characteristics of each field and add it to the File Definition. All attributes of those five files have a text type, because there is no possibility of performing any computations on those attributes.

To maintain data integrity, we may attach a validation formula to some fields. A validation formula may be thought of as a rule that regulates what can be entered into the field. Superbase-4 provides several types of validation such as functions for cross-file checking, a range limit for imposing an upper and lower limit for numeric fields, and lists

of acceptable values for text fields. After we define the characteristics of all the attributes and add validation formulas to the appropriate fields, the next step is to specify the index fields. For each file, at least one field must be specified as an index field. We must also create an index for any field which will be used for relational operations such as a join operation. For example, for the Faculty file there are three fields assigned as an index: FAC_ID, LASTNAME and D_CODE. These fields are used as relation links between Faculty file and Publication, Offering, Course or Department files. FAC_ID is a unique index. The data in this field must not be repeated in any other record. LASTNAME and D_CODE are duplicate indexes and allow duplicate entries.

5. Designing The Forms

Designing a form to process each menu selection is the most difficult part of developing the system. Designing a form in the proposed system is more than just placing text and data on the form. The interface between a form and a set of DML subroutines must also be constructed because all functions of the ESIS are controlled by the DML program. To construct the interface, we attached the DML commands to the pushbuttons that are located on the form.

To put the users at ease and help them sort out the contents of the screen, we used different colors for the forms of different menus, and we also used different fonts and different character sizes for menu titles, items and instructions.

a. The Main Menu Form

The Main menu form contains the name of the system and seven pull-down menus, namely Faculty, Course, Offering, Department, Update, Report and Others (see Figure 3.3). The system always returns to the Main menu form after completing a process.

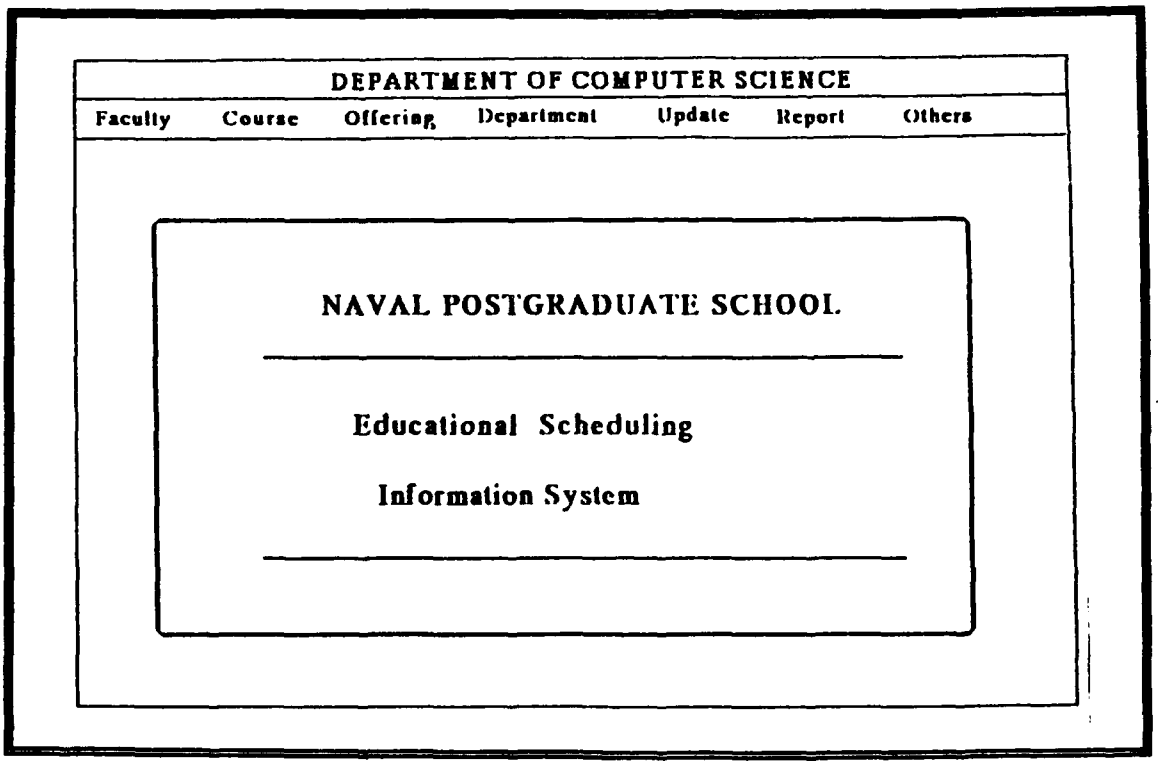
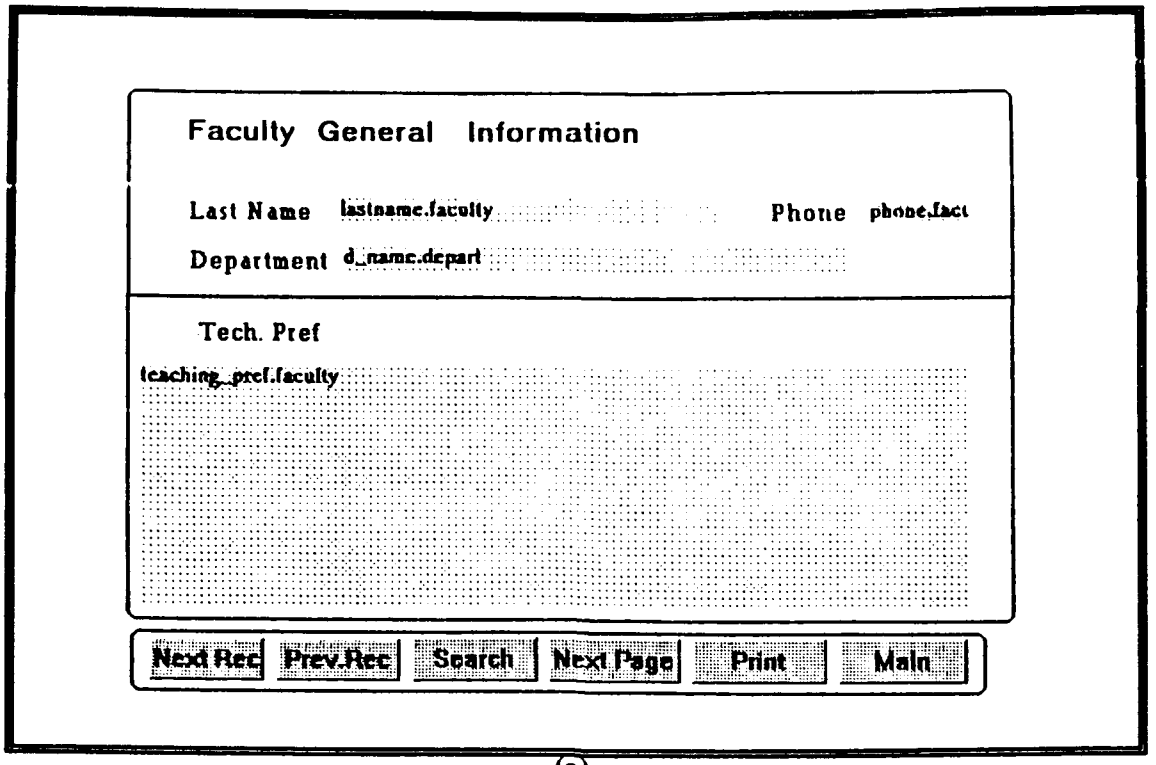


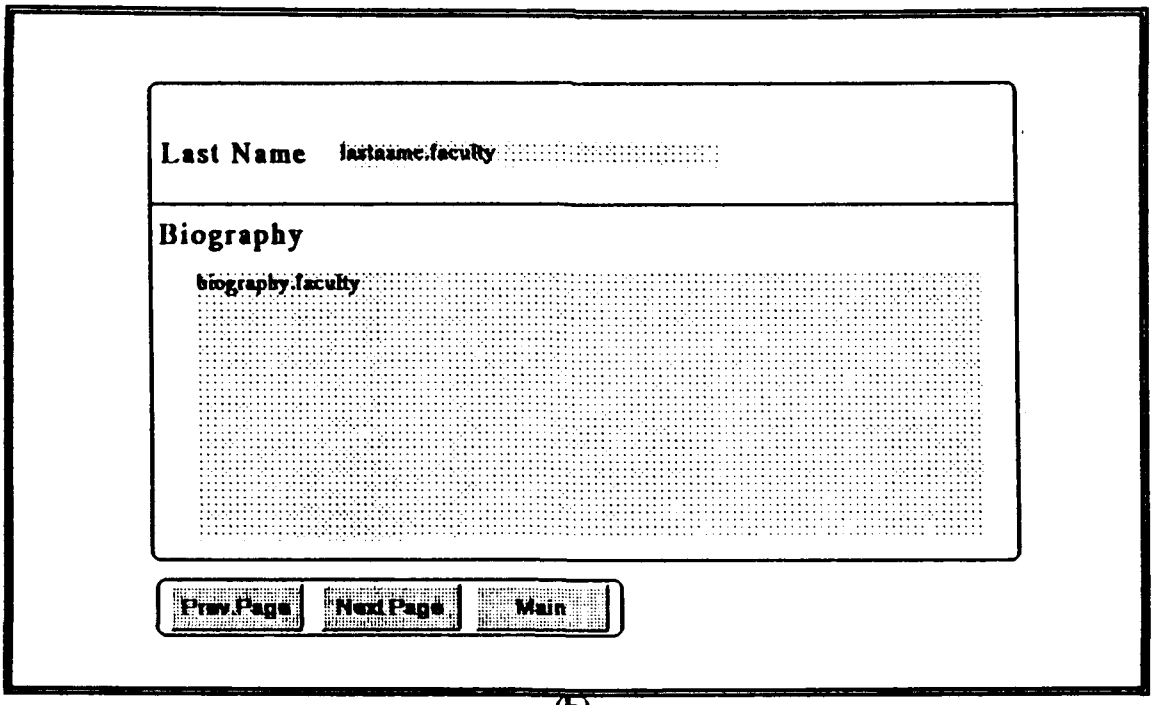
Figure 3.3 The Main Menu

b. The Faculty Menu Forms

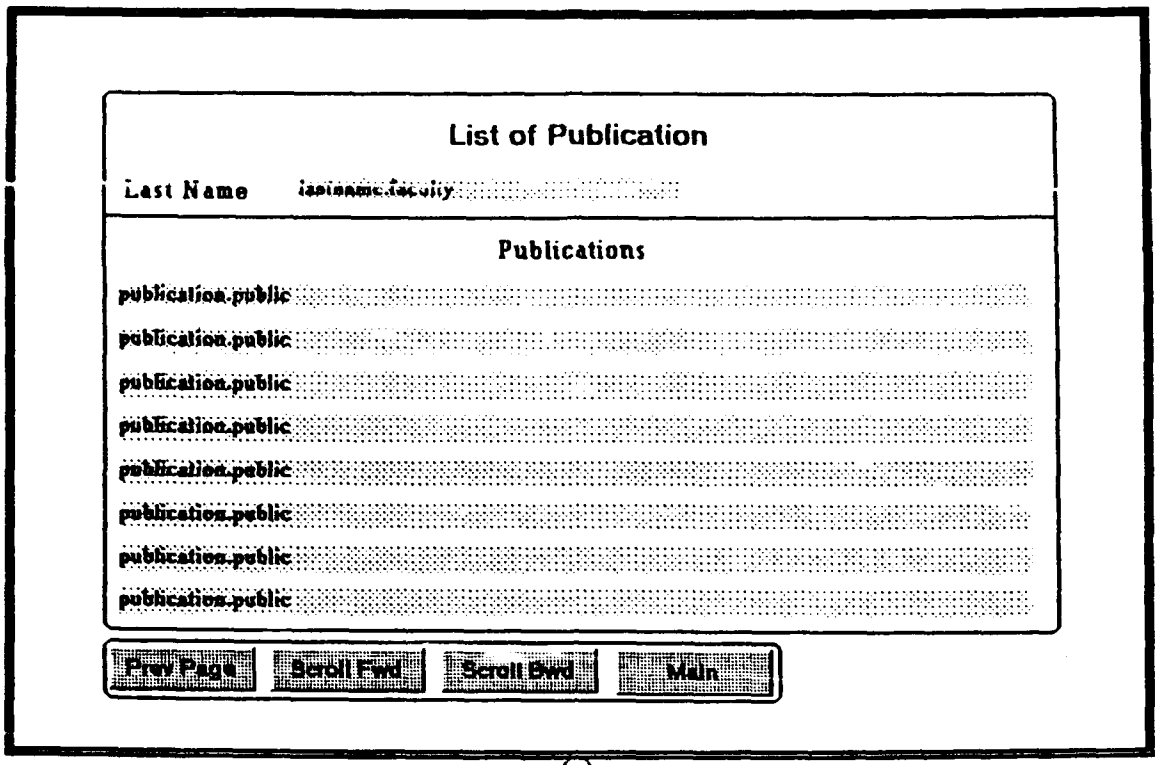
This group consists of three forms: Faculty General Information, Course Assignment and Publication forms. The Faculty General Information form provides general information about faculty members such as: last name, phone number, teaching preference, biography and list of publications. We use a multi-page form to display all of this information (Figure 3.4). The transaction line function of Form Designer is used to display multiple records of the Publication file on a single page. To go from one page to another page we may click the NEXT PAGE or PREV PAGE pushbuttons. It is possible to find a specific record from the Faculty file using the SEARCH pushbutton.


Faculty General Information
Last Name lastname.faculty
Phone phone.fact
Department d_name.depart
Tech. Pref
teaching_pref.faculty
Next Rec
Prev.Rec
Search
Next Page
Print
Main

(a)


Last Name lastname.faculty
Biography
biography.faculty
Prev.Page
Next Page
Main

(b)



(c)

Figure 3.4 The Faculty General Information Form

The Course Assignment form is designed to display the last name of a faculty member with the list of courses assigned to him or her (Figure 3.5). This form illustrates the one-to-many relation between the Faculty file and the Course file. Eight records in the Course file can be displayed at once.

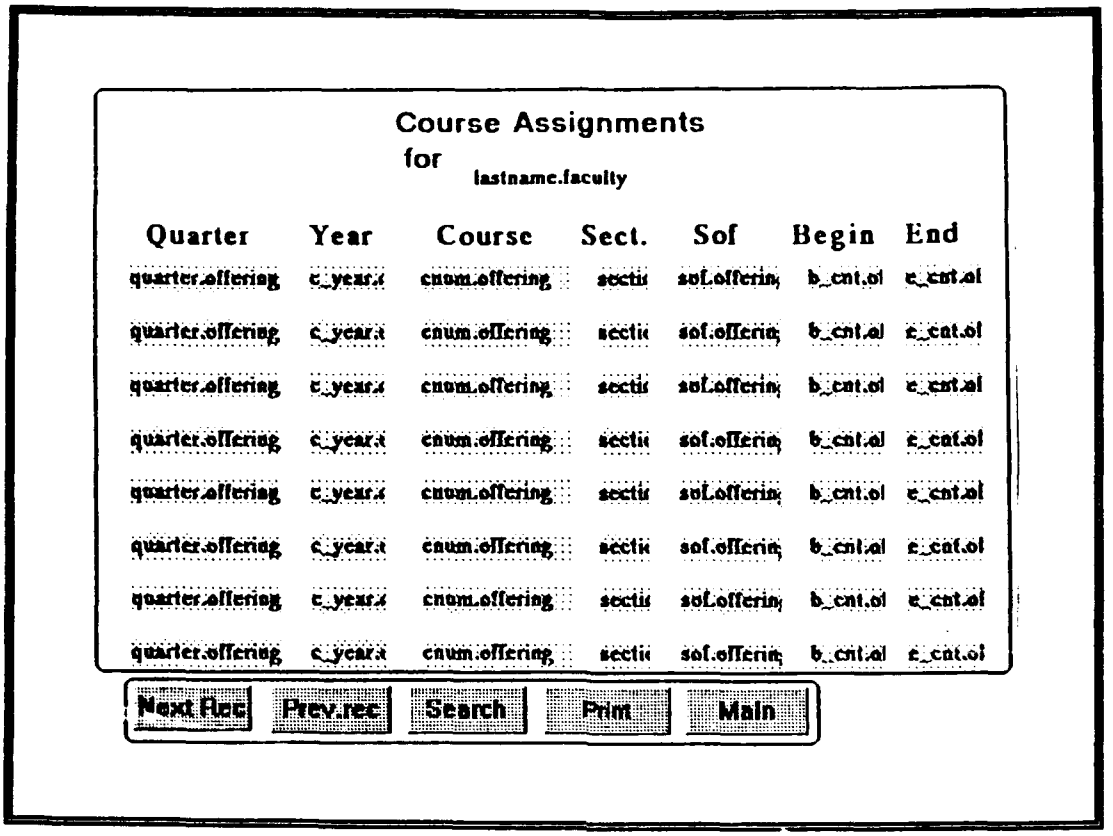


Figure 3.5 The Course Assignment Form

The Publication form is designed to provide information about journals written by faculty members (Figure 3.6). The information includes the name of the publication, journal title, authors, volume, number and number of pages. We may use the SEARCH pushbutton to find a record of Publication file by authors.

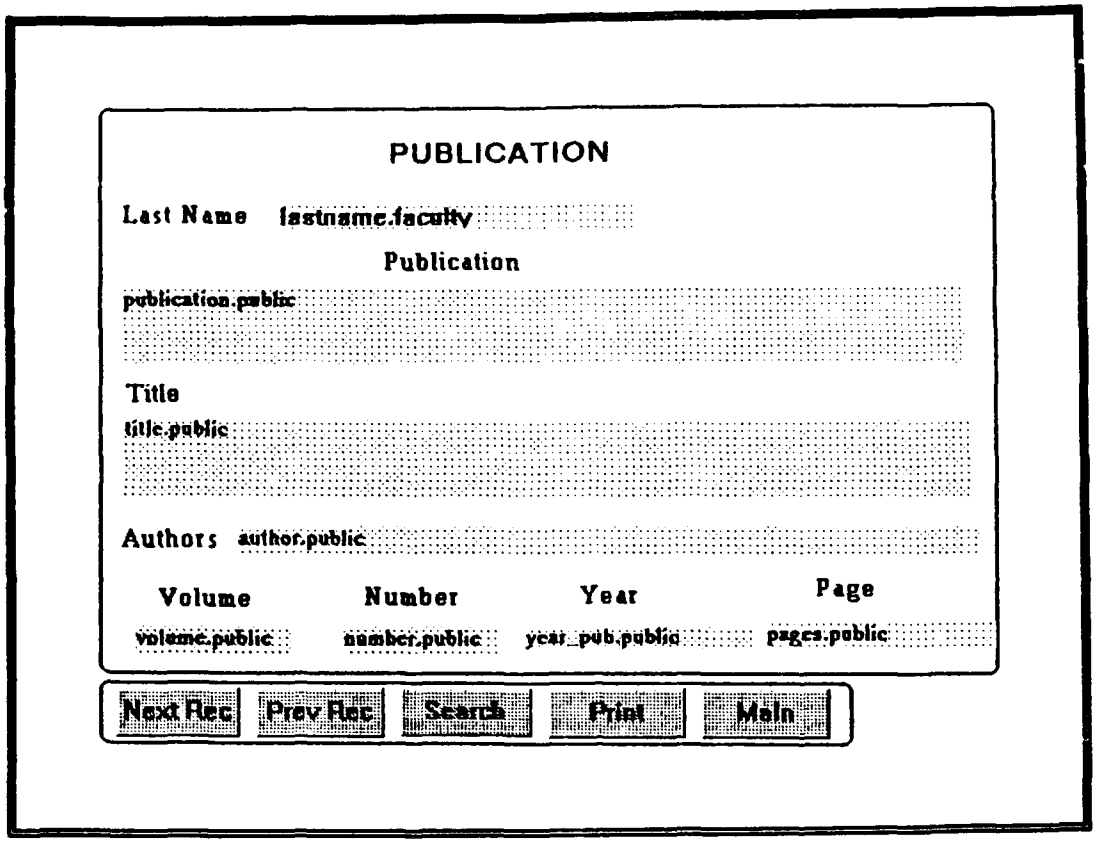
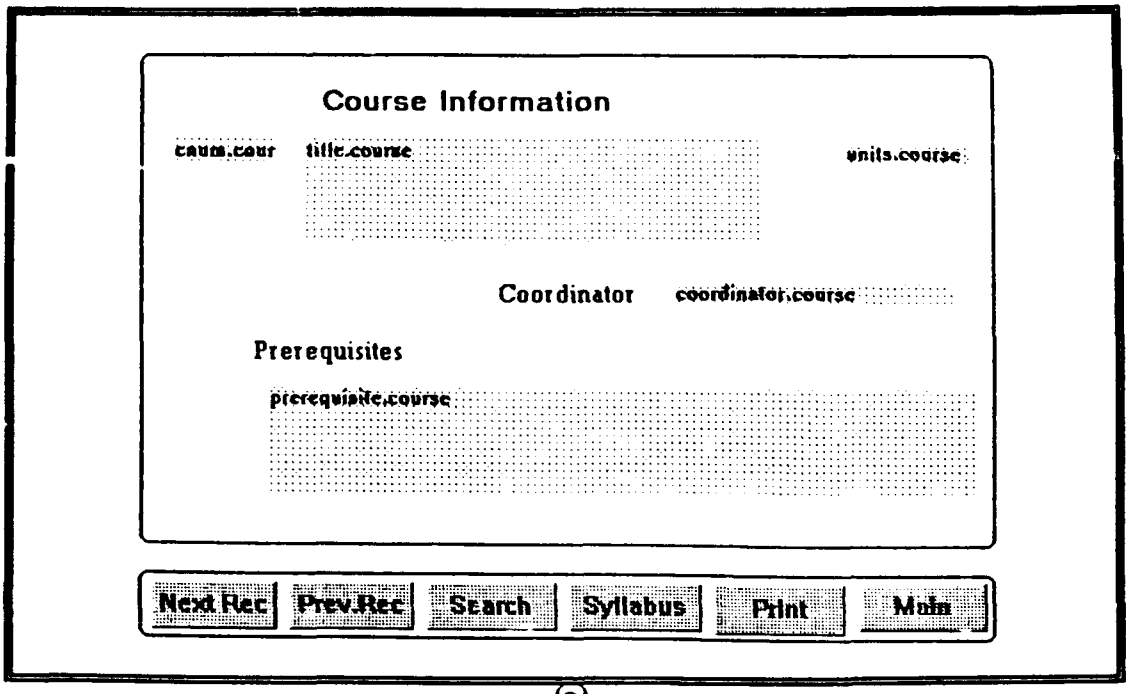


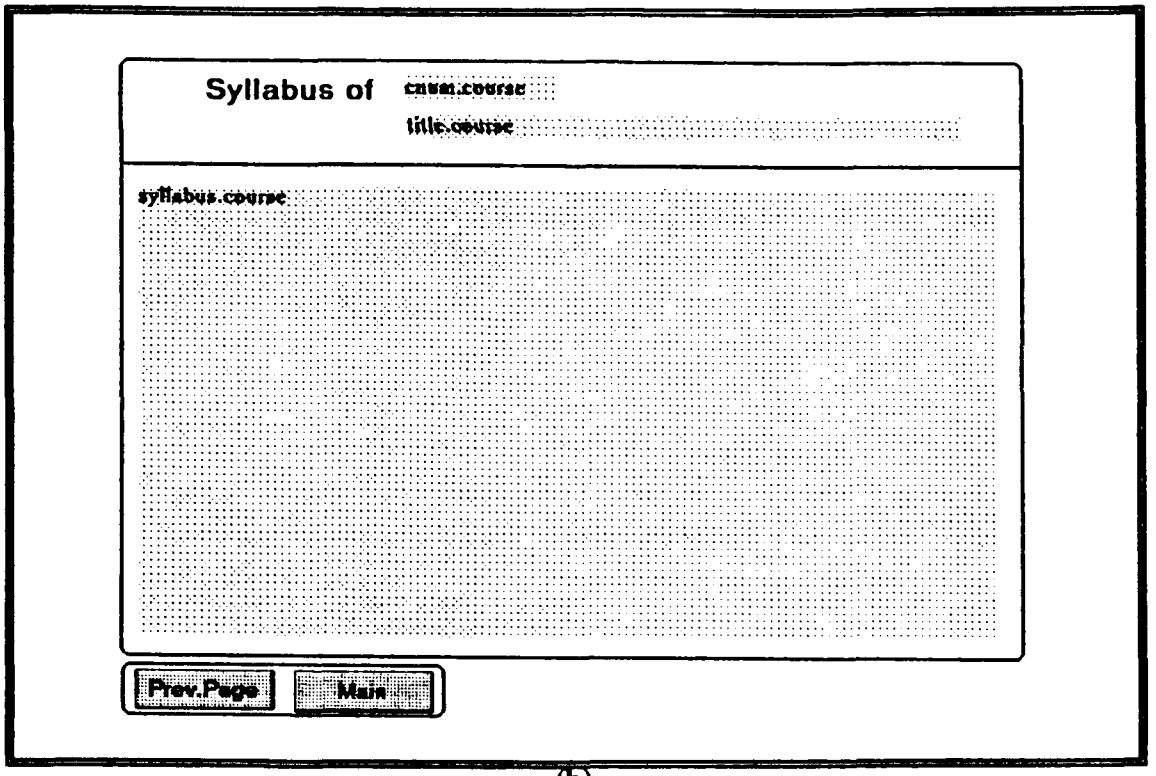
Figure 3.6 The Publication Form

c. The Course Menu Forms

This group consist of two forms: Course General Information form and Faculty Assignment form. The Course General Information form provides the information such as: course code, course title, credit, coordinator, prerequisites and syllabus. All of this information is displayed in two pages (Figure 3.7). The first page displays course code, course title, credit, coordinator and prerequisites. The second page displays the syllabus.



(a)



(b)

Figure 3.7 The Course General Information Form

The Faculty Assignment form is designed to display the course title and the list of faculty members who are teaching the course (Figure 3.8). Ten faculty members in the Faculty file can be displayed at once using the one-to-many replication features of Form Designer.

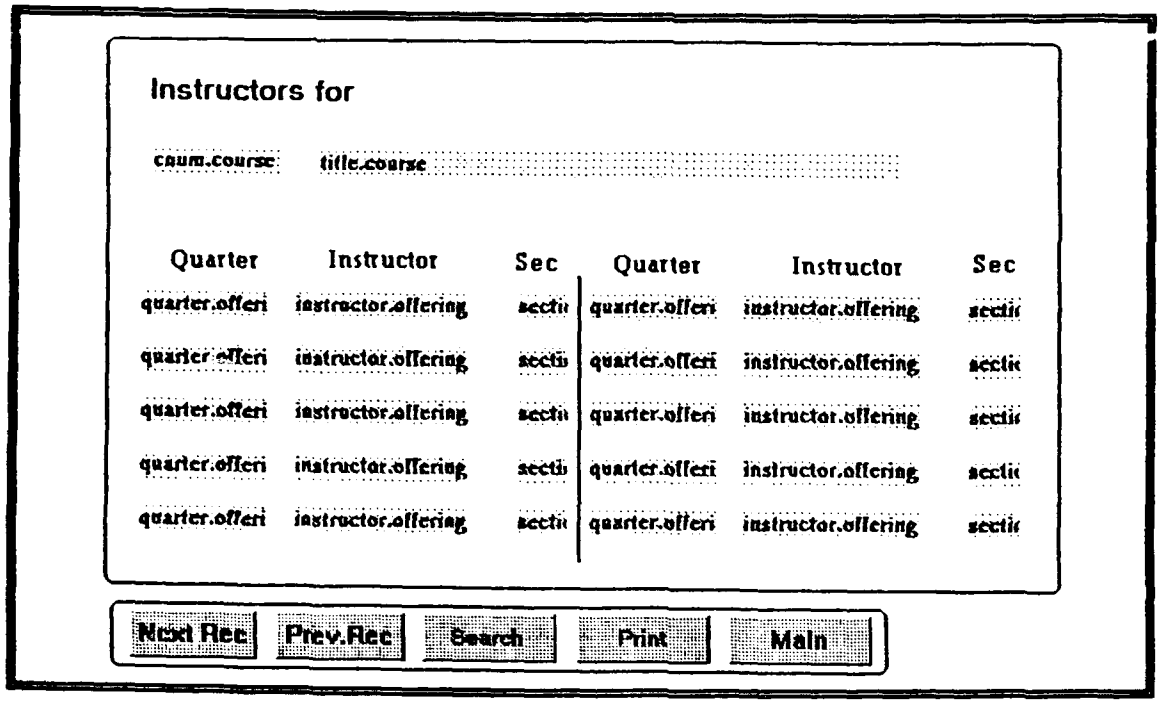


Figure 3.8 The Faculty Assignment Form

d. The Offering Menu Form

The only form for the Offering menu is the Offering General Information form. A single record in the Offering file is displayed on a single page (Figure 3.9). Five pushbuttons are available for this form: PREV REC, NEXT REC, SEARCH, PRINT and MAIN. SEARCH pushbutton can be used to find a special record from the Offering file using a course code as a key.

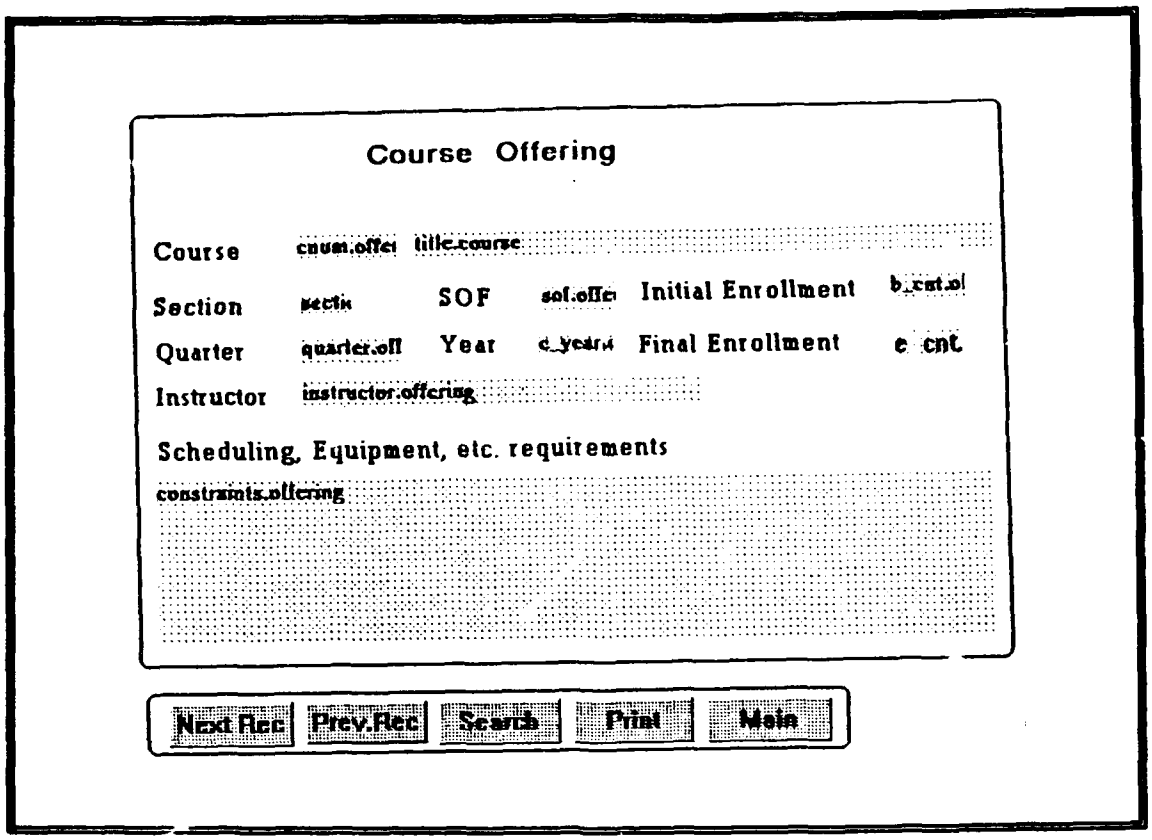


Figure 3.9 The Offering Form

e. The Department Menu Forms

This group consists of the Department General Information form and the Department Faculty form. The Department General Information form provides information about the department such as: department code, department name, chairman and phone number (Figure 3.10). The Department Faculty form provides information on the faculty members who work in one department (Figure 3.11).

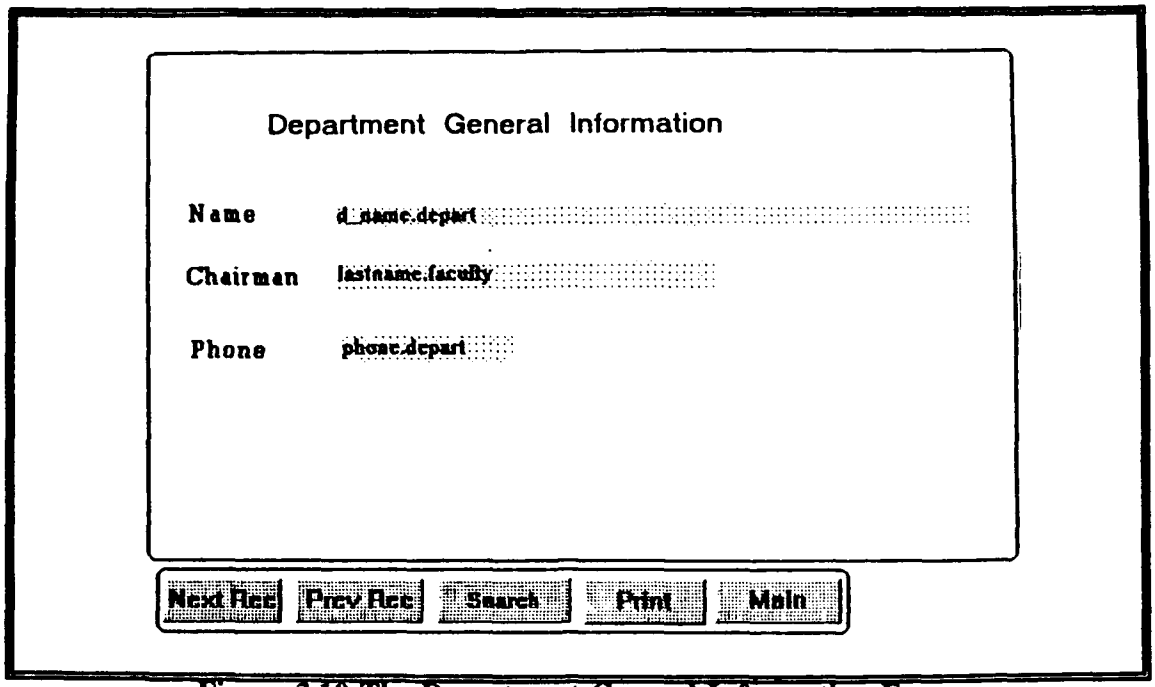


Figure 3.10 The Department General Information Form

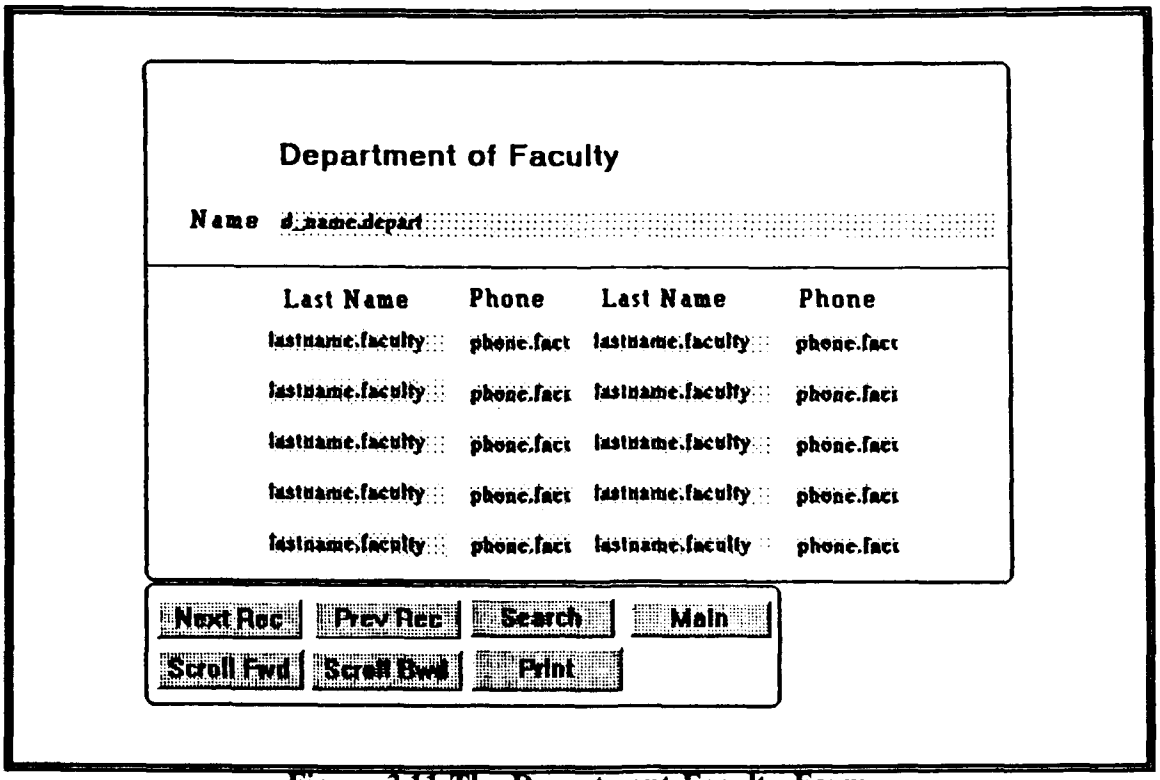


Figure 3.11 The Department Faculty Form

The Department General information form is designed to display a single record of the Department file on a single page. This is simpler than the Department Faculty form which is designed to display multiple records from the Faculty file in a single page. SCROLL FWD and SCROLL BWD pushbuttons are available in the Department Faculty form to enable browsing forward and backward in the Faculty file.

f. The Update Menu Forms

This group consists of updating routines that must be carried out from time to time in various files. The two submenus provided are for modifying and inserting records. The form of each submenu contains five pushbuttons with the labels FACULTY, COURSE, DEPARTMENT, OFFERING and PUBLICATION to indicate the file being updated, and one pushbutton with the label CANCEL to indicate canceling the update process and returning to the main menu (Figure 3.12 and Figure 3.13).

Each submenu consists of five forms: the form for Faculty, Course, Department, Offering and Publication files. For every form there is a pushbutton with the label UNDO. This pushbutton is used to cancel the update process or the insert process that we have done. When we select a Modify submenu for a particular file, the dialog box will automatically be displayed by the REQUEST function of Superbase-4. The dialog box contains a list of the values of primary keys for all records in the file. To find a record for modification, we may search from the list and highlight it.

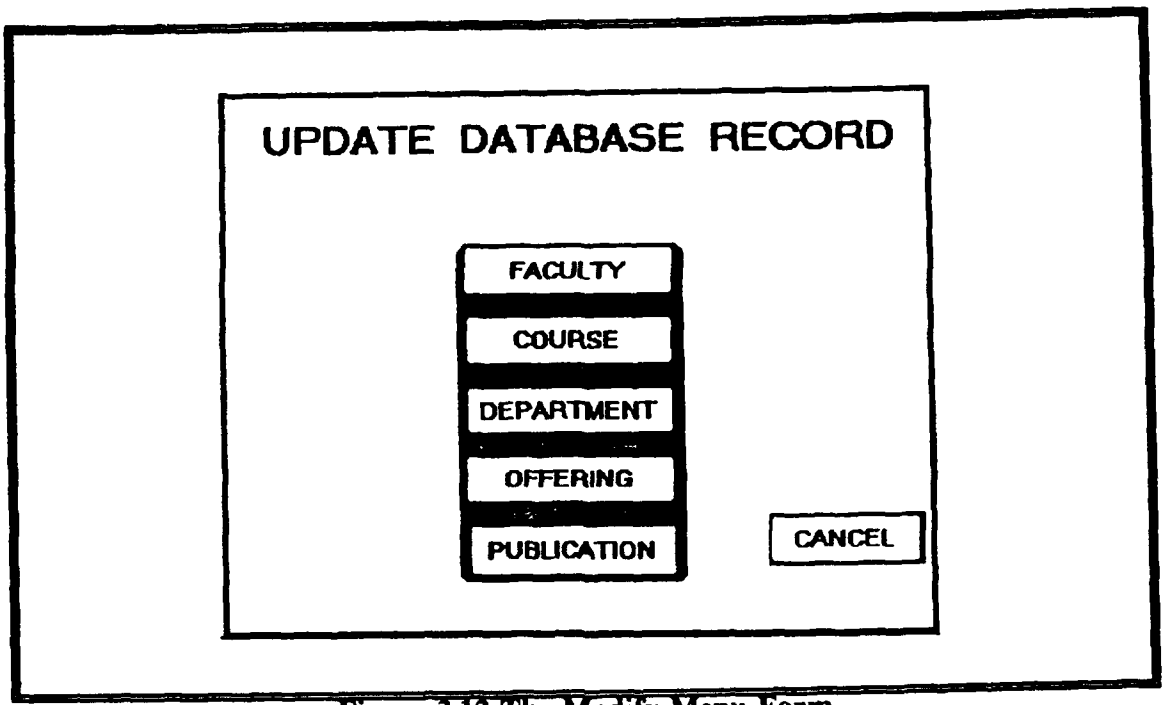


Figure 3.12 The Modify Menu Form

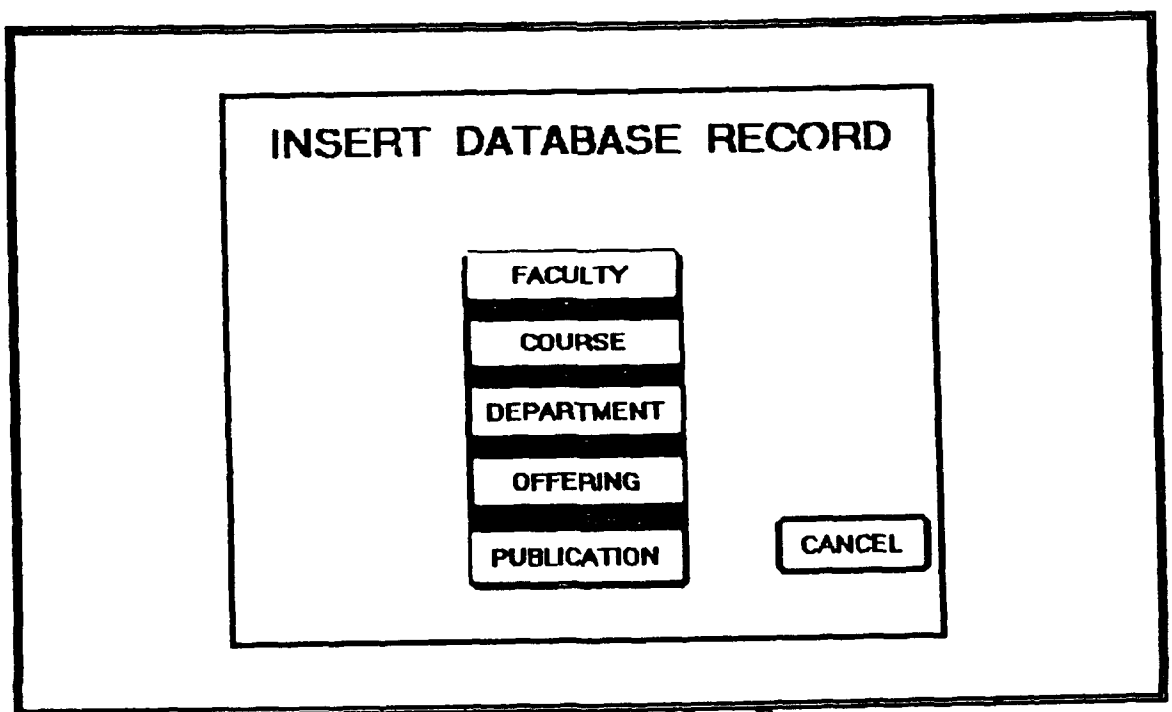


Figure 3.13 The Insert Menu Form

g. The Report Menu

This group consists of two reports that have been predefined for the ESIS: Faculty Teaching Assignment report and Course Assignment report. The first report is designed to print all faculty members with their assigned courses. The second report is designed to print all courses with the list of faculty members who are teaching each course.

To make the reports easy to read, we group the Faculty Teaching Assignment report by the faculty member's last name and the Course Assignment report by course code. All of these reports are made using the report facility of the form designer.

h. The Others Menu

This is the last menu of the ESIS. It contains two options: help and exit. The help option provides assistance to the users by explaining the menu structure of the system and how to run the system. The exit option is for quitting the system.

6. Coding the DML Programs

As we stated earlier, all functions of the ESIS are controlled by the DML programs. Once the main program is running, the system retains program control of the user's access to Superbase's functions and facilities until the exit option is selected from the main menu.

There are a total of twenty three programs in the system (see Table 3.1). Each program consists of several routines and each routine consists of several DML commands. Each program controls the execution of one form. The interface between a program and its form is constructed by attaching the commands to the pushbutton. Each pushbutton on the form is handled by one routine in the program. When a given program is executed, the related form will be opened. Only one form can be opened at a time. The Form Designer opens all

files that are needed by the form. When we click one of the pushbuttons that are available on the form, the routine for the selected pushbutton will be executed.

TABLE 3.1 THE ESIS PROGRAMS

Prog.Name	Purpose
MAIN.SBP	- Clear the Superbase menu system. - Display Main form with seven pull-down menus. - Set on the Superbase menu system when Exit option is selected.
FAC_G_I.SBP	- Display the multi-pages Faculty General Information form. - Handle the NEXT REC, PREV REC, NEXT PAGE, PREV PAGE, SEARCH, PRINT and MAIN pushbuttons.
FAC_C_I.SBP	- Display the Faculty Course Assignment form. - Handle the NEXT REC, PREV REC, SEARCH, PRINT and MAIN pushbuttons.
PUBLIC.SBP	- Display the Publication form - Handle the NEXT REC, PREV REC, SEARCH, PRINT and MAIN pushbuttons.

CRS_G_I.SBP	- Display the multi-pages Course General Information form. - Handle the NEXT REC, PREV REC, SEARCH, PREV PAGE, SYLLABUS, PRINT and MAIN pushbuttons.
CRS_I_A.SBP	- Display the Course Assignment form. - Handle the NEXT REC, PREV REC, SEARCH, PRINT and MAIN pushbuttons.
OFR_G_I.SBP	- Display the Offering General Information form. - Handle the NEXT REC, PREV REC, SEARCH, PRINT and MAIN pushbuttons.
DEP_G_I.SBP	- Display the Department General Information form. - Handle the NEXT REC, PREV REC, SEARCH, PRINT and MAIN pushbuttons.
DEP_I_F.SBP	- Display the Department Faculty form. - Handle the NEXT REC, PREV REC, SCROLL FWD, SCROLL BWD, SEARCH, PRINT and MAIN pushbuttons.
I_MAIN.SBP and U_MAIN.SBP	- I_MAIN.SBP displays the Insert submenu form. - U_MAIN.SBP displays the Modify submenu form. - Both handles the FACULTY, COURSE, DEPARTMENT,OFFERING, PUBLICATION and CANCEL pushbuttons.

I_FAC.SBP, I_COURSE.SBP, I_OFFER.SBP, I_PUBLIC.SBP and I_DEPT.SBP	- I_FAC inserts the FACULTY record. - I_COURSE inserts the COURSE record. - I_OFFER inserts the OFFERING record. - I_PUBLIC inserts the PUBLICATION record. - I_DEPT inserts the DEPARTMENT record. - All programs handle the INSERT, UNDO and MAIN pushbuttons.
U_FAC.SBP, U_COURSE.SBP, U_OFFER.SBP, U_PUBLIC.SBP and U_DEPT.SBP	- U_FAC modifies the FACULTY record. - U_COURSE modifies the COURSE record. - U_OFFER modifies the OFFERING record. - U_PUBLIC modifies the PUBLICATION record. - U_DEPT modifies the DEPARTMENT record. - All programs handle the MODIFY, UNDO and MAIN pushbuttons.
C_SCH_RP.SBP	- Prints the faculty teaching assignment report. - The report is grouped by faculty's last name and quarter.
I_SCH_RP.SBP	- Prints the course assignment report. - The report is grouped by Course code and quarter.

Two programs have been designed to print the reports. C_SCH_RP.SBP is designed to print the faculty teaching assignment and I_SCH_RP.SBP to print the course assignment. These programs are automatically generated by the Report Generator of the Form Designer when we save the report.

From the experience of developing a simple form-based application such as the ESIS, we found some advantages as well as disadvantages of using the form-based approach. In the next chapter we will evaluate the application development process using a form-based approach.

IV. EVALUATION

In the previous chapter we described the design of ESIS and discussed how we employed the features of Superbase-4. Now we will evaluate the application development process using a form-based approach. We will evaluate two form-oriented methods for database design, the relational category of Superbase-4, and the advantages and disadvantages of using the Form Designer.

A. FORM-ORIENTED DATABASE DESIGN

One of the most difficult tasks in database design is properly collecting the relevant information [Ref. 10:p. 161]. There are two categories of information needed in database design, information about the processes and information about the data. Database design requires detailed knowledge not only about the characteristics of the data but also about the existing and projected processes that operate on the data.

There are two methods that can be used to design a database in a form-oriented application development: E/R Diagram-Relational-Forms method and Forms-Relational method.

1. E/R Diagram-Relational-Forms Method

Our implementation uses this method. First we identify the objects the application needs. An object is the representation of an entity in the user's work environment, such as a course, faculty or department. We must identify the objects and their structures so that we may determine what data must be stored in the database. To learn what objects are

important to the user and to identify the manner in which those objects are processed, we must communicate with the end-users.

When all of the objects have been identified, we may determine the relationship between those objects. Relationships can be one-to-one, one-to-many or many-to-many. After all relationships have been determined, we may draw an Entity/Relationship diagram (E/R diagram). An E/R diagram is an abstract database design and an illustration of the logical structure of a database.

From an E/R diagram, we may then construct a relational database by mapping all its entities and some relationships into relations. Each entity type is mapped into a base relation. Each many-to-many relationship and multi-valued attribute are mapped into the base relation. Let us take an example from the ESIS. This system consist of three objects: faculty, course and department. The database thus contains three base relations: FACULTY, COURSE and DEPARTMENT. Since the offering relationship (associates faculty members and courses) is a many-to-many relationship, then we also have an OFFERING base relation. Publication is a multi-valued attribute of faculty entity, therefore we can map this attribute to a PUBLICATION relation.

When the database design has been finished, the last step of this process is form design. Communication between users and designers is important during the form design process.

The E/R Diagram-Relational-Forms method gives the user and developer great flexibility in determining the design of the forms. Users and developers design the forms they need based on the underlying relational database that is already designed. Unfortunately this method is time consuming and labor intensive, because we must set up the

specifications by interviewing the users. Ambiguities, misunderstanding, and over-specification or under-specification of an application may occur.

2. Forms-Relational Method

The difference between this method and the previous method is in the way we capture the essential information for designing the database. Using the E/R Diagram-Relational-Forms method we capture the information by interviewing the users while in the Forms-Relational method we capture the information by identifying the forms that the users already have.

Each form maps into a base relation. For example, in the ESIS we have five forms: faculty, course, department, offering and publication forms, thus we also have five relations: FACULTY, COURSE, DEPARTMENT, OFFERING and PUBLICATION. Each form consist of several data items. These data items are the attributes of the relation.

To determine the relationship between relations, we must identify the common fields of the associated forms. For example, in the faculty form we have data item C_NUM (C_NUM is a field name of the course number). This data item also exists in the course form, therefore we determine that there must also exist a relationship between FACULTY and COURSE relations.

Designing a database using the second method is not a difficult task, since all information needed for database design has been provided on the form. The designer does not spend too much time trying to filter out the nonessential information. The problem with this method is that when the users need to add new forms or delete existing forms, the database structure must be modified. Thus, the Form-Relational method gives less flexibility

to the user to modify the system than the E/R Diagram-Relational-Form method. This is the reason we chose the first method for designing a database in our proposed system.

B. THE RELATIONAL CATEGORY OF SUPERBASE-4

A relational database consists of three major parts: a structural part, an integrity part and a manipulation part [Ref. 3:p. 369]. The structure part consists essentially of n-ary relations, and nothing else. The integrity part consists of two general integrity rules, namely "entity integrity" and "referential integrity". The entity integrity rule states that no component of the primary key of a relation is allowed to accept nulls [Ref. 3:p. 279]. The referential integrity rule states that the values of a given foreign key must match values of the corresponding primary key [Ref. 3:p. 282]. And finally, the manipulation part provides a set of algebraic operators for data manipulation.

In 1982, Codd proposed that a database management system be regarded as relational only if it supports the operations restrict, project and join, without requiring any predefinition of physical access paths to support those operations [Ref. 3:p. 376]. Superbase-4 therefore, does not qualify as a truly relational system, since in Superbase-4 we may perform a join operation according to values of some field only if that field is indexed.

Joins are time-consuming and expensive operations. Because of this Superbase-4 uses an indexing technique to make joins less expensive to the system and its users. The only problem with this technique is that index files need more memory space, especially for a large database.

Furthermore, Codd divides the relational DBMS into four categories based on the degree of supporting the aspect of the relational model: tabular, minimally relational, relationally complete and fully relational.

Tabular is a system that supports tables as a data structure but not the set of algebraic operators. Minimally relational is a system that supports tables, restrict, project and join but no other relational operators. Relationally complete is a system that supports tables and all of the operators of the relational algebra. Finally, fully relational is a system that support all aspects of the model. Superbase-4 falls in the second category, that is minimally relational because Superbase-4 only supports tables, restrict and project using the SELECT command, and join using the Link command.

C. THE ADVANTAGES AND DISADVANTAGES OF USING SUPERBASE-4

The goal of this thesis is to study the effectiveness of a form-based interface. The form approach was considered the most natural interface between end-user and data because a large number of end-users employ forms or version of forms in their daily work activities [Ref. 10:p. 162]. Even though end-users may be unfamiliar with programming and query languages, they can still perform complex operations on predefined forms through simple interactive interfaces provided by the form systems.

Several office information systems based on the forms concept have been designed and implemented. For example, Office-By-Example (OBE) provides a screen interface to an underlying relational database. FADS, a Form Application and Development System designed at the University of California at Berkeley, includes facilities for the interactive development of form applications and provides an interface to the INGRES database system. FORMANAGER have been developed by General Research Board of the University of

Maryland, includes facilities for form specification, form processing, and form control. [Ref. 12:pp. 238-239]. Precision Software Limited presents a Form Designer with facilities for the interactive development of form applications and provides an interface to the relational database system [Ref. 8:p. 1-1].

In this section we present the advantages and disadvantages of using the Form Designer of Superbase-4.

1. Advantages

a. Easy to Use

The Form Designer is designed to generate forms used in database applications. A form is specified interactively using the Form Designer toolbox. The structural layout of the form is defined using a full screen editor, and the fields are defined through a dialog box. This approach has an advantage. By using a full-screen editor, we can view the form exactly as it will be used. Interactive specification allows the system to assist the user by prompting the required information and giving immediate warning for errors.

The Form Designer toolbox is easy to use because the toolbox uses the direct manipulation approach to control the user command. The direct manipulation approach means that the operations or manipulations that are legal at any time are displayed for the user. The user is not required to memorize the commands. Instead he selects the appropriate command from among the choices displayed.

The Form Designer is different from the screen painter of dBASE IV. Although the screen painter of dBASE IV allows users to interactively create forms for entering and editing data, there is a fundamental difference between dBASE IV forms and Form Designer forms. dBASE IV forms may not include commands or programs, while Form Designer

forms can. This difference makes the dBASE IV forms more difficult to use by the end-users. The dBASE IV forms need a program to control the execution.

b. Easy to Modify

One difficulty in designing interactive systems is that the user may not have a clear idea of what the system should be when it is done. For this reason the designers must have a thorough understanding of the diverse community of users and the task that must be accomplished. A clearer understanding of user preferences can be helpful in designing the system. The software with easy-to-use facilities to design the screen format, such as the Form Designer of Superbase-4, facilitates adjusting to the changing needs of the user.

The Form Designer of Superbase-4 has an easy-to-use facility to design the screen format, namely the toolbox. The toolbox allows us to select an object on the form, then change its appearance, move or resize it. Text objects may be edited, either by changing the characteristic of the object, or by changing the text itself. Changing files and fields references are also permitted.

The Form Designer forms are easy to modify. They are designed separately from the DML programming language and we may modify the forms without changing the program related to the forms. This approach is different from some other form systems such as the Office Procedure Automation System (OPAS) from IBM research group [Ref. 15:p. 328].

In OPAS, a form is considered as an information holding object consisting of two parts. The first part is a form heading which provides the form name, form structure and component names. The second part consists of one or more form instances. Forms processing is performed via a high-level programming language, called FORMAL (from

Forms ORiented MAnipulation Language). Each process takes one or more forms as input and produces a single form as output. Each process specification contains: 1) a title line that specifies the name of the form, processes to be performed and the name of the output form, 2) a form heading for the output form, 3) a description of the data constituting the form and 4) qualification for the intended process, which may include the source of data or the condition to be applied in selecting form instances. This approach makes form modifications more complicated than the form modifications in the Form Designer of Superbase-4 because every modification of the OPAS forms will affect the programs related to the forms.

c. Self-Contained Command

The Form Designer includes facilities for adding commands to forms. Commands may be attached to pushbuttons or placed directly on the form as command objects. When commands are attached to a pushbutton, they will be executed when we click the pushbutton. When commands are put on the form as command objects, they will be executed if the insertion point passes through them.

By having a self-contained command facility, we may create a complete application without writing a DML program. Commands such as OPEN FORM, CLOSE FORM or FORM [SHOW] [page #] can be attached to pushbutton and may be used to control the application.

2. The Disadvantages

a. Difficult to maintain

If we have a self-contained command facility then we may create an application without writing any DML program in the traditional sense. However, for more complex applications we need to combine the form's built in processing facility with a program.

Since the forms are designed separately from the DML program environment, we need to construct an interface between a form and a set of DML subroutines. An interface can be constructed by attaching the keywords RUN, CHAIN, GOTO, LOAD or GOSUB to the pushbuttons.

The problem arises when later the user wants to modify the application. Since the interface commands are attached to the pushbuttons, physically they are separate from the DML program, although logically they are not. To modify the application, the user must consider both sides, the interface commands and the DML subroutines, and make sure that the modification does not cause a conflict between them.

To eliminate the problems of maintenance, the Form Designer must be able to handle all operations that are needed by the application without any DML program. The program is used only to control the execution order of the forms. This approach has been used by the FORMANAGER from the University of Maryland [Ref. 12:p. 240].

Similar to the Form Designer, the FORMANAGER specifies the forms interactively. The specification process involves two stages: 1) define the form syntax; and 2) define the form semantics. The specification of a field involves defining the field types and field actions. The field type defines the usage of the field in the form application, such as search field, display field and entry field. The field action describes the way in which a user can perform an action on the underlying database system, such as input action or update action. FORMANAGER translates all insert and update actions to the appropriate SQL commands on the underlying database. The forms are executed by mapping the specifications from the form into queries on the underlying database.

When we have an application with a number of interrelated forms, we may construct a procedural control of order and the conditions under which the forms are executed using a high-level procedural language such as one developed by IBM research [Ref. 12:p. 256]. This approach is different from that of Form Designer.

In Form Designer, the DML program is used not only to handle the order of form executions but also to handle the complicated form operations that can not be handled by Form Designer such as transaction lines browsing. Thus in Form Designer the form operations are done by both the DML program and Form Designer. While in the FORMANAGER, the form operations are handled by FORMANAGER itself. The program is needed when we want to control the execution order of the interrelated forms.

b. The Message Dialog

The wording of messages and the layout of information on a screen are important for novice users [Ref. 9:p. 312]. Densely packed screens may overwhelm the beginner. With only modest effort, screen formats can be substantially improved to reduce search time. Multiple windows are helpful for users, but displays that are cluttered are distracting.

Superbase-4 provides the message dialogs to pass the messages from the system to the user and vice versa. There are two groups of message dialogs: Superbase's own format, and those that use the standard Windows message dialogs. Keyword REQUEST may be used to select one of the Superbase's dialogs and display it on the screen. The format and the location of the message dialog on the screen are defined by the system, and may not be changed by the application designer. Unfortunately the placement of the message dialog box in the middle of the screen, can be distracting to the user.

To improve the benefits of using the message dialogs in the Form Designer, the application designer should be able to determine the location, the size or the format of the message dialog based on the application needs. Otherwise, the application designers should create their own message dialog especially when the message is used as a warning to the user for verifying the data on the screen that are already typed by the user.

c. Transaction Lines

Many applications include one-to-many relationship between records in different files. For example, the relationship between the DEPARTMENT and FACULTY relations is one-to-many. This means that each record of the DEPARTMENT relation relates to more than one record of the FACULTY relation. To display such a relationship, the Form Designer provides transaction lines. The term "transaction lines" refers to a group of fields of the "many" part of a relationship. Using the above example, the transaction lines refers to a group of fields of FACULTY relation.

The problem of using transaction lines is that we cannot include more than one one-to-many relationship. For example, suppose we want to create a form for DEPARTMENT, FACULTY and COURSE relations. Since the relationship between DEPARTMENT and FACULTY is one-to-many and the relationship between FACULTY and OFFERING is also one-to-many, we are not able to create those relations in one form.

In comparison, FORMANAGER allows more than one one-to-many relationship in transaction lines. Fields for repeating groups are denoted by field names followed by a period and a digit which indicates the level of repetition. To make the Form Designer of Superbase-4 a better tool, it must contain complete user facilities for all aspects of application development for a relational database system, including the facility for handling

a nested repeating group. The nested repeating group facility can be used to reduce the number of forms in the ESIS and make the system simpler. For example, the Department Faculty form under the Department menu and Course Assignment form under the Faculty menu can be combined into one form. This form will contain the information about the faculty members who work in a given department and a list of courses that are assigned to each faculty member.

V. CONCLUSION

Due to the improvements in database technology and the widespread dependence on database management systems, users with little database experience must frequently interact with an organizational database. Users need a user-friendly interface to interact with database systems. The relational query languages such as SQL and QUEL are not ideal languages for end-users. The forms approach is considered the most natural interface between end-user and database because a large number of end-users employ forms in their daily work activities. Several systems based on the forms concept have been designed and implemented, such as OBE, FADS, OPAS, FORMANAGER and Form Designer of Superbase-4.

To study the effectiveness of a form-base interface, we implemented a form-based application named The Educational Scheduling Information System (ESIS) using the Form Designer of Superbase-4. An evaluation of our implementation indicates that designing a user interface using a form-based approach is easier than designing a user interface using relational query languages. The Form Designer of Superbase-4 has easy-to-use facilities for creating the forms. The facilities are based on the direct manipulation approach. The Form Designer forms are different than forms from other systems. Using the self-contained command facility of the Form Designer, we may have applications with several interrelated forms without writing a DML program.

There are still some issues that need to be considered in future research. One of these issues is determining alternative methods of designing relations in form-based applications. Another issue is to investigate methods to overcome the limitations of the Form Designer.

Two methods of designing relations in form-based applications have been discussed. Each method has advantages and disadvantages but the E/R Diagram-Relational-Forms method is more suitable than the Forms-Relational method. The E/R Diagram-Relational-Forms method gives flexibility to the users and application developers. The users and application developers are free to design the forms they need based on the underlying relational database. This method is time consuming and labor intensive.

Several limitations of the Form Designer were found during the development of ESIS. The command interface and the DML program environment are not closely integrated. The message dialog is located inappropriately. The transaction lines facility places an arbitrary limit of one line at a time. We made some recommendations for improvement in these areas.

APPENDIX A. RELATIONAL DATABASE SCHEMA

CNUM	TITLE	UNIT	COORDI NATOR	DESCRIP TION	PREREQ	SYLLABUS

(a). Course Relation

FAC_ID	LAST NAM E	FIRST NAME	PHONE	TEACHING_ PREF	D_CODE	BIOGR APHY

(b). Faculty Relation

CNUM	INST RUCT OR	QUAR TER	SECT ION	CONST RAINT S	BEGIN_ CNT	END_ CNT	SOF	C_YE AR

(c). Offering Relation

D_CODE	D_NAME	CHAIRMAN	PHONE

(d). Department Relation

F_ID	PUBLIC ATION	AUTHOR	TITLE	VOL	NO	YEAR	PAGES

(e). Publication Relation

APPENDIX B. DML SOURCE CODE LISTING

The following listings are the DML code that was created or modified for the Educational Scheduling Information System.

```
        REM ------------------------------------------
        REM         MAIN
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author   : C.Thomas Wu
        REM         Created  : June 20, 1990
        REM         Modified : August 20, 1990 by Partoyo
        REM ------------------------------------------
        REM Erase the system menu and display the opening      REM screen
        MENU CLEAR
        OPEN FORM "open_scr"

Lstart:
        REM  Set the heading
        MENU CLEAR
        hD$ = DATE$ (TODAY ,"mmm dd, yy")
        hT$ = TIME$ (NOW ,"hh:mm  am")
        hH$ = hD$ + "    DEPARTMENT OF COMPUTER SCIENCE     " +
hT$
        SET HEADING hH$

        REM Define the menu
        MENU 1,0,1,"&Faculty"
        MENU 1,1,1,"&General Info"
        MENU 1,2,1,"Course &Assignment"
        MENU 1,3,1,"Publication"

        MENU 2,0,1,"&Course"
        MENU 2,1,1,"&General Info"
        MENU 2,2,1,"&Faculty &Assignment"

        MENU 3,0,1,"&Offering"
```

```
        MENU 3,1,1,"General Info"

        MENU 4,0,1,"&Department"
        MENU 4,1,1,"&General Info"
        MENU 4,2,1,"&Department &Faculty"

        MENU 5,0,1,"&Update"
        MENU 5,1,1,"&New Record"
        MENU 5,2,1,"&Modify"

        MENU 6,0,1,"&Report"
        MENU 6,1,1,"By &Instructor"
        MENU 6,2,1,"By &Course"

        MENU 7,0,1,"&Others"
        MENU 7,1,1,"&Help"
        MENU 7,2,1,"&Exit"

Lprocess:
        PANEL OFF
        MENU ON mnu%,sub_menu%
        WAIT MENU
        MENU CLEAR
        ON mnu% GOTO fac,crs,ofr,dep,upd,rep,ext

fac:
        SELECT CASE sub_mnu%
        CASE 1
                CHAIN "fac_g_i"
        CASE 2
                CHAIN "fac_c_a"
        CASE 3
                CHAIN "fac_pub"
        END SELECT

crs:
        SELECT CASE sub_mnu%
        CASE 1
                CHAIN "crs_g_i"
        CASE 2
                CHAIN "crs_i_a"
        END SELECT
```

```
ofr:
        CHAIN "ofr_g_i"

dep:
        SELECT CASE sub_mnu%
        CASE 1
                CHAIN "dep_g_i"
        CASE 2
                CHAIN "dep_i_f"
        END SELECT

upd:
        SELECT CASE sub_mnu%
        CASE 1
                CHAIN "i_main"
        CASE 2
                CHAIN "u_main"
        END SELECT

rep:
        SELECT CASE sub_mnu%
        CASE 1
                CHAIN "i_sch_rp"
        CCASE 2
                CHAIN "c_sch_rp"
        END SELECT
        GOTO Lstart
ext:
        IF sub_mnu% = 1 THEN
                REQUEST "   Help System","dummy",139
                GOTO Lstart
        ELSE
                MENU CLEAR :CLOSE ALL :PANEL ON : SET HEADING "": END
        END IF


        REM ------------------------------------------
        REM             FAC_C_A
        REM             Dept of Computer Science
        REM             Departemental Database
        REM
        REM             Author   : C.Thomas Wu
```

```
        REM         Created  : June 20, 1990
        REM         Modified : August 20, 1990 by Partoyo
        REM ----------------------------------------------

        ON ERROR GOTO L99
        PANEL OFF
        OPEN FORM "inst_sch"
        INDEX lastname

L1:     REM Wait here for a pushbutton to be clicked
        FORM
        WAIT MOUSE
        GOTO L1

L11:    REM Next Record
        SELECT FORM NEXT
        ON ERROR GOTO L21
        GOTO L1

L12:    REM Prev Record
        SELECT FORM PREVIOUS
        ON ERROR GOTO L21
        GOTO L1

L13:    REM Select a specific record by Fac.last name
        REQUEST "Select a Faculty's lastname",
              "",20,a%,a$,15,lastname.faculty
        SELECT FORM FIRST
        IF a% = 0 THEN GOTO L1
        WHILE lastname.faculty <> a$
              SELECT FORM NEXT
        WEND
        GOTO L1

L14:    REM Print the form
        PRINT CURRENT PAGE USING 1,0,0,0,1,0,1
        GOTO L1

L15:    REM Exit and return to aminawenu
        MENU CLEAR
        CHAIN "main"
```

```
L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another selection",
                  1,a%
      IF a% = 1 THEN RESUME L1
      CHAIN "main"
      END


      REM ------------------------------------------
      REM         FAC_G_I
      REM         Dept of Computer Science
      REM         Departemental Database
      REM
      REM         Author   : C.Thomas Wu
      REM         Created  : June 21, 1990
      REM         Modified : August 24, 1990 by Partoyo
      REM ------------------------------------------

      ON ERROR GOTO L99
      PANEL OFF
      OPEN FORM "faculty"
      INDEX lastname

L1:   REM Wait here for a pushbutton to be clicked
      FORM
      WAIT MOUSE
      GOTO L1

L11:  REM Next Record
      SELECT FORM NEXT
      ON ERROR GOTO L21
      GOTO L1

L12:  REM Prev Record
      SELECT FORM PREVIOUS
      ON ERROR GOTO L21
      GOTO L1
```

```
L15:  REM Exit and return to main menu
      MENU CLEAR
      CHAIN "main"

L16:  REM Print the form
      PRINT CURRENT PAGE USING 1,0,0,0,1,0,1
      GOTO L1

L17:  REM Select a specific record by Fac.last name
      REQUEST "Select a Faculty's lastname",
            "",20,a%,a$,15,lastname.faculty
      SELECT FORM FIRST
      IF a% = 0 THEN GOTO L1
      WHILE lastname.faculty <> a$
            SELECT FORM NEXT
      WEND
      GOTO L1

L1b:  REM Brwosing foward
      IF RECCOUNT("*") <> 0 THEN
            SELECT FORM NEXT PAGE
      ELSE
            SELECT FORM CURRENT
      END IF
      GOTO L1

L1c:  REM Browsing backward
      IF RECCOUNT ("*") <> 0 THEN
            SELECT FORM PREVIOUS PAGE
      ELSE
            SELECT FORM CURRENT
      END IF
      GOTO L1

L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another
            selection",1,a%
```

```
IF a% = 1 THEN RESUME L1
CHAIN "main"
END


REM ------------------------------------------
REM         FAC_PUB
REM         Dept of Computer Science
REM         Departemental Database
REM
REM         Author   : Partoyo
REM         Created  : August 21, 1990
REM         Modified :
REM ------------------------------------------

ON ERROR GOTO L99
PANEL OFF
OPEN FORM "public"
INDEX lastname

L1:   REM Wait here for a pushbutton to be clicked
      FORM
      WAIT MOUSE
      GOTO L1

L11:  REM Next Record
      SELECT NEXT
      ON ERROR GOTO L21
      GOTO L1

L12:  REM Prev Record
      SELECT PREVIOUS
      ON ERROR GOTO L21
      GOTO L1

L13:  REM Exit and return to amin menu
      MENU CLEAR
      CHAIN "main"

L14:  REM Select a specific record by Fac.last name
      REQUEST "Select Author",  "",20,a%,a$,50,author.public
      SELECT FORM FIRST
```

```
        IF a% = 0 THEN GOTO L1
        WHILE author.public <> a$
              SELECT FORM NEXT
        WEND
        GOTO L1

L15:    REM Print the form
        PRINT CURRENT PAGE USING 1,0,0,0,1,0,1
        GOTO L1

L21:    REM Panel errors
        REQUEST ERR$ ( ERRNO ), " ",2,a%
        SELECT CURRENT
        GOTO L1

L99:    REM Error conditions
        REQUEST ERR$ ( ERRNO ),"press OK to make another
              selection",1,a%
        IF a% = 1 THEN RESUME L1
        CHAIN "main"
        END


        REM ------------------------------------------------
        REM             CRS_G_I
        REM             Dept of Computer Science
        REM             Departemental Database
        REM
        REM             Author   : C.Thomas Wu
        REM             Created  : June 21, 1990
        REM             Modified : August 5,1990 by Partoyo
        REM ------------------------------------------------

        ON ERROR GOTO L99
        PANEL OFF
        OPEN FORM "course"

L1:     REM Wait here for a pushbutton to be clicked
        FORM
        WAIT MOUSE
        GOTO L1
```

```
L11:  REM Next Record
      SELECT NEXT
      ON ERROR GOTO L21
      GOTO L1

L12:  REM Prev Record
      SELECT PREVIOUS
      ON ERROR GOTO L21
      GOTO L1

L14:  REM Exit and return to amin menu
      MENU CLEAR
      CHAIN "main"

L17:  REM Select a specific record by Fac.last name
      REQUEST "Select Course Code",    "",20,a%,a$,6,cnum.course
      SELECT FIRST
      IF a% = 0 THEN GOTO L1
      WHILE cnum.course <> a$
            SELECT NEXT
      WEND
      GOTO L1

L18:  REM Print the form
      PRINT CURRENT PAGE ALL USING 1,0,0,0,1,0,1
      DISPLAY
      FORM 1
      GOTO L1

L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another
            selection",1,a%
      IF a% = 1 THEN RESUME L1
      CHAIN "main"
      END
```

```
REM ------------------------------------------
REM         CRS_I_A
REM         Dept of Computer Science
REM         Departemental Database
REM
REM         Author   : C.Thomas Wu
REM         Created  : June 21, 1990
REM         Modified : August 7,1990 by Partoyo
REM ------------------------------------------

ON ERROR GOTO L99
PANEL OFF
OPEN FORM "crse_sch"

L1:   REM Wait here for a pushbutton to be clicked
      FORM
      WAIT MOUSE
      GOTO L1

L11:  REM Next Record
      SELECT NEXT
      ON ERROR GOTO L21
      GOTO L1

L12:  REM Prev Record
      SELECT PREVIOUS
      ON ERROR GOTO L21
      GOTO L1

L13:  REM Exit and return to main menu
      MENU CLEAR
      CHAIN "main"

L14:  REM Select a specific record by Title.course
      REQUEST "Select Course
      Title","",20,a%,a$,50,title.course
      SELECT FORM FIRST
      IF a% = 0 THEN GOTO L1
      WHILE title.course <> a$
            SELECT FORM NEXT
      WEND
      GOTO L1
```

```
L15:  REM Print the form
      PRINT CURRENT PAGE ALL USING 1,0,0,0,1,0,1
      DISPLAY
      GOTO L1

L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another
    selection",1,a%
      IF a% = 1 THEN RESUME L1
      CHAIN "main"
      END


      REM ------------------------------------------
      REM         OFR_G_I
      REM         Dept of Computer Science
      REM         Departemental Database
      REM
      REM         Author   : C.Thomas Wu
      REM         Created  : June 21, 1990
      REM         Modified : August 24,1990 by Partoyo
      REM ------------------------------------------

      ON ERROR GOTO L99
      PANEL OFF
      OPEN FORM "offering"

L1:   REM Wait here for a pushbutton to be clicked
      FORM
      WAIT MOUSE
      GOTO L1

L11:  REM Next Record
      SELECT FORM NEXT
      ON ERROR GOTO L21
      GOTO L1
```

```
L12:  REM Prev Record
      SELECT FORM PREVIOUS
      ON ERROR GOTO L21
      GOTO L1

L13:  REM Exit and return to main menu
      MENU CLEAR
      CHAIN "main"

L14:  REM Select a specific record by cnum.offering
      REQUEST "Select Course
            Number","",20,a%,a$,14,cnum.offering
      SELECT FORM FIRST
      IF a% = 0 THEN GOTO L1
      WHILE title.course <> a$
            SELECT FORM NEXT
      WEND
      GOTO L1

L15:  REM Print the form
      PRINT CURRENT PAGE ALL USING 1,0,0,0,1,0,1
      DISPLAY
      GOTO L1

L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another
    selection",1,a%
      IF a% = 1 THEN RESUME L1
      CHAIN "main"
      END


      REM ------------------------------------------
      REM         DEP_G_I
      REM         Dept of Computer Science
      REM         Departemental Database
      REM
```

```
        REM         Author   : Partoyo
        REM         Created  : August 24, 1990
        REM         Modified :
        REM -----------------------------------------

        ON ERROR GOTO L99
        PANEL OFF
        OPEN FORM "dep_scr"

L1:     REM Wait here for a pushbutton to be clicked
        FORM
        WAIT MOUSE
        GOTO L1

L11:    REM Next Record
        SELECT FORM NEXT
        ON ERROR GOTO L21
        GOTO L1

L12:    REM Prev Record
        SELECT FORM PREVIOUS
        ON ERROR GOTO L21
        GOTO L1

L13:    REM Exit and return to main menu
        MENU CLEAR
        CHAIN "main"

L14:    REM Select a specific record by Department
        REQUEST "Select Department Name","",
                20,a%,a$,50,d_name.depart
        SELECT FORM FIRST
        IF a% = 0 THEN GOTO L1
        WHILE d_name.depart <> a$
                SELECT FORM NEXT
        WEND
        GOTO L1

L15:    REM Print the form
        PRINT CURRENT PAGE ALL USING 1,0,0,0,1,0,1
        DISPLAY
        GOTO L1
```

```
L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another
            selection",1,a%
      IF a% = 1 THEN RESUME L1
      CHAIN "main"
      END


      REM ------------------------------------------
      REM         DEP_I_F
      REM         Dept of Computer Science
      REM         Departemental Database
      REM
      REM         Author   : Partoyo
      REM         Created  : August 8, 1990
      REM         Modified :
      REM ------------------------------------------

      ON ERROR GOTO L99
      PANEL OFF
      OPEN FORM "work_scr"

L1:   REM Wait here for a pushbutton to be clicked
      FORM
      WAIT MOUSE
      GOTO L1

L11:  REM Next Record
      SELECT FORM NEXT
      ON ERROR GOTO L21
      GOTO L1

L12:  REM Prev Record
      SELECT FORM PREVIOUS
      ON ERROR GOTO L21
      GOTO L1
```

```
L13:  REM Next Page
      IF RECCOUNT ("*") <> 0 THEN
            SELECT FORM PREVIOUS PAGE
      ELSE
            SELECT FORM CURRENT
      END IF
      GOTO L1

L14:  REM Previous Page
      IF RECCOUNT ("*") <> 0 THEN
            SELECT FORM PREVIOUS PAGE
      ELSE
            SELECT FORM CURRENT
      END IF
      GOTO L1

L15:  REM Exit and return to main menu
      MENU CLEAR
      CHAIN "main"

L16:  REM Select a specific record by Department
      REQUEST "Select Department Name","",
            20,a%,a$,50,d_name.depart
      SELECT FORM FIRST
      IF a% = 0 THEN GOTO L1
      WHILE d_name.depart <> a$
            SELECT FORM NEXT
      WEND
      GOTO L1

L17:  REM Print the form
      PRINT CURRENT PAGE ALL USING 1,0,0,0,1,0,1
      DISPLAY
      GOTO L1

L21:  REM Panel errors
      REQUEST ERR$ ( ERRNO ), " ",2,a%
      SELECT CURRENT
      GOTO L1

L99:  REM Error conditions
      REQUEST ERR$ ( ERRNO ),"press OK to make another
```

```
        selection",1,a%
IF a% = 1 THEN RESUME L1
CHAIN "main"
END


REM ---------------------------------------------
REM          I_SCH_RP
REM          Dept of Computer Science
REM          Departemental Database
REM
REM          Author   : Partoyo
REM          Created  : August 24, 1990
REM          Modified :
REM ---------------------------------------------

OPEN FILE "C:\SUPBASE\THESIS\faculty"
OPEN FILE "C:\SUPBASE\THESIS\THESIS\offering"

ON ERROR GOTO errt

REQUEST "Report to Printer ?","",134,a%
IF a% = 0 THEN END ELSE IF a% = 1 THEN PRINT;
REPORT

HEADING
?
? @12;"   DEPARTMENT OF COMPUTER SCIENCE"
? @12;"FACULTY TEACHING ASSIGMENT 1990 - 1991"
? @25;&15 TODAY
END HEADING

GROUP lastname.faculty
BEFORE GROUP lastname.faculty
? @7;&4fac_id.faculty;@14;&15lastname.faculty
END GROUP

AFTER GROUP lastname.faculty
?
END GROUP

GROUP qurater.offering
```

```
        BEFORE GROUP quarter.offering
        ? @34;&7quarter.offering
        ?
        END GROUP

        AFTER GROUP quarter.offering
        ?
        END GROUP

        SELECT @42;&6cnum.offering;@51;&1section.offering
        WHERE lastname.faculty = instructor.offering
        ORDER lastname.faculty,quarter.offering

getout:     REM back to main program
        CHAIN "main"

errt:   REM error condition
        RESUME getout
        REM ------------------------------------------------
        REM         C_SCH_RP
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author   : Partoyo
        REM         Created  : August 24, 1990
        REM         Modified :
        REM ------------------------------------------------

        OPEN FILE "C:\SUPBASE\THESIS\course"
        OPEN FILE "C:\SUPBASE\THESIS\THESIS\offering"

        ON ERROR GOTO errt

        REQUEST "Report to Printer ?","",134,a%
        IF a% = 0 THEN END ELSE IF a% = 1 THEN PRINT;
        REPORT

        HEADING
        ?
        ? @12;"DEPARTMENT OF COMPUTER SCIENCE"
        ? @12;" COURSE ASSIGNMENT 1990 - 1991"
        ? @25;&15 TODAY
```

```
        END HEADING

        GROUP cnum.course
        BEFORE GROUP cnum.course
        ?
        ? @5;&6cnum.course;@13;&54title.course
        ?
        END GROUP

        AFTER GROUP cnum.course
        ?
        END GROUP

        GROUP qurater.offering
        BEFORE GROUP quarter.offering
        ? @39;&7quarter.offering
        ?
        END GROUP

        AFTER GROUP quarter.offering
        ?
        END GROUP

        SELECT @48;&15instructor.offering
        WHERE cnum.course = cnum.offering
        ORDER cnum.course,quarter.offering

getout:         REM back to main program
        CHAIN "main"

errt:   REM error condition
        RESUME getout


        REM ----------------------------------------
        REM         U_MAIN
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author   : Partoyo
        REM         Created  : September 8, 1990
        REM         Modified :
```

```
        REM ------------------------------------------
        REM Erase the system menu and display the opening screen
        MENU CLEAR
        OPEN FORM "u_main"

L1:     REM wait here for a pushbutton to be clicked
        FORM
        WAIT MOUSE
        GOTO L1

LUF:    REM Update Faculty
        CHAIN "u_fac"

LUC:    REM Update Course File
        CHAIN "u_course"

LUD:    REM Update Department
        CHAIN "u_dept"

LUO:    REM Update Offering
        CHAIN "u_offer"

LUP:    REM Update Publication
        CHAIN "u_public"

EXT:    REM back to main program
        CHAIN "main"


        REM ------------------------------------------
        REM         U_FAC
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author   : Partoyo
        REM         Created  : September 20, 1990
        REM         Modified :
        REM ------------------------------------------
        REM Erase the system menu and display the blank form

        ON ERROR GOTO L99
        ms1$ = "Is entry correct ?"
```

```
        ms2$ = "Select another record ?"
        ms3$ = "Save current record ?"
        OPEN FORM "u_fac"

L11:    REM Select record
        REQUEST "Select Faculty Last Name",""
                        20,a%,a$,15,lastname.faculty
        SELECT FIRST
        IF a% = 0 THEN GOTO L12
        WHILE lastname.faculty <> a$
                SELECT NEXT
        WEND
        FORM

L111:REM Update the existing record
        MOUSE ON
        ENTER
        MOUSE OFF

L112:REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L111
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN
                STORE
                GOTO L11
        ELSE
                MOUSE ON
                GOTO L12
        END IF

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        REQUEST ms3$,"",130,a%
        IF a% = 1 THEN STORE
        CHAIN "main"

L14:    REM Undo adding record
        SELECT CURRENT
        FORM
```

```
        GOTO L111

L99:    REM Error condition
        REQUEST ERR$ ( ERRNO )," Press OK to make another
                selesction",1,a%
        IF a% = 1 THEN GOTO L11
        GOTO L12
        END


        REM ------------------------------------------------
        REM             U_COURSE
        REM             Dept of Computer Science
        REM             Departemental Database
        REM
        REM             Author   : Partoyo
        REM             Created  : September 20, 1990
        REM             Modified :
        REM ------------------------------------------------
        REM Erase the system menu and display the blank form

        ON ERROR GOTO L99
        ms1$ = "Is entry correct ?"
        ms2$ = "Select another record ?"
        ms3$ = "Save current record ?"
        OPEN FORM "u_course"

L11:    REM Select record
        REQUEST "Select Course Code",""
                        20,a%,a$,6,cnum.course
        SELECT FIRST
        IF a% = 0 THEN GOTO L12
        WHILE cnum.course <> a$
                SELECT NEXT
        WEND
        FORM

L111:REM Update the existing record
        MOUSE ON
        ENTER
        MOUSE OFF
```

```
L112:REQUEST ms1$,"",130,a%
     IF a% = 0 THEN GOTO L111
     REQUEST ms2$,"",130,a%
     IF a% = 1 THEN
          STORE
          GOTO L11
     ELSE
          MOUSE ON
          GOTO L12
     END IF

L12: REM Wait here for a pushbutton to be clicked
     WAIT MOUSE
     GOTO L12

L13: REM Back to the main program
     REQUEST ms3$,"",130,a%
     IF a% = 1 THEN STORE
     CHAIN "main"

L14: REM Undo adding record
     SELECT CURRENT
     FORM
     GOTO L111

L99: REM Error condition
     REQUEST ERR$ ( ERRNO )," Press OK to make another
          selesction",1,a%
     IF a% = 1 THEN GOTO L11
     GOTO L12
     END


     REM ------------------------------------------
     REM          U_OFFER
     REM          Dept of Computer Science
     REM          Departemental Database
     REM
     REM          Author   : Partoyo
     REM          Created  : September 20, 1990
     REM          Modified : October 12,1990
     REM ---------- ----------------------------------
```

```
        REM Erase the system menu and display the blank form

        ON ERROR GOTO L99
        ms1$ = "Is entry correct ?"
        ms2$ = "Select another record ?"
        ms3$ = "Save current record ?"
        OPEN FORM "u_offer"

L11:    REM Select record
        REQUEST "Select Course Code",""
                        20,a%,a$,6,cnum.offering
        SELECT FIRST
        IF a% = 0 THEN GOTO L12
        WHILE cnum.offering <> a$
                SELECT NEXT
        WEND
        FORM

L111:REM Update the existing record
        MOUSE ON
        ENTER
        MOUSE OFF

L112:REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L111
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN
                STORE
                GOTO L11
        ELSE
                MOUSE ON
                GOTO L12
        END IF

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        REQUEST ms3$,"",130,a%
        IF a% = 1 THEN STORE
        CHAIN "main"
```

```
L14:  REM Undo adding record
      SELECT CURRENT
      FORM
      GOTO L111

L99:  REM Error condition
      REQUEST ERR$ ( ERRNO )," Press OK to make another
            selesction",1,a%
      IF a% = 1 THEN GOTO L11
      GOTO L12
      END


      REM ------------------------------------------
      REM          U_DEPT
      REM          Dept of Computer Science
      REM          Departemental Database
      REM
      REM          Author   : Partoyo
      REM          Created  : September 20, 1990
      REM          Modified : October 12,1990
      REM ------------------------------------------
      REM Erase the system menu

      ON ERROR GOTO L99
      ms1$ = "Is entry correct ?"
      ms2$ = "Select another record ?"
      ms3$ = "Save current record ?"
      OPEN FORM "u_dept"

L11:  REM Select record
      REQUEST "Select Department Name",""
                  20,a%,a$,50,d_name.depart
      SELECT FIRST
      IF a% = 0 THEN GOTO L12
      WHILE d_name.depart <> a$
            SELECT NEXT
      WEND
      FORM

L111:REM Update the existing record
      MOUSE ON
```

```
        ENTER
        MOUSE OFF

L112:REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L111
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN
                STORE
                GOTO L11
        ELSE
                MOUSE ON
                GOTO L12
        END IF

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        REQUEST ms3$,"",130,a%
        IF a% = 1 THEN STORE
        CHAIN "main"

L14:    REM Undo adding record
        SELECT CURRENT
        FORM
        GOTO L111

L99:    REM Error condition
        REQUEST ERR$ ( ERRNO )," Press OK to make another
                selesction",1,a%
        IF a% = 1 THEN GOTO L11
        GOTO L12
        END


        REM ------------------------------------------
        REM             U_PUBLIC
        REM             Dept of Computer Science
        REM             Departemental Database
        REM
        REM             Author   : Partoyo
```

```
        REM         Created  : September 20, 1990
        REM         Modified : October 12,1990
        REM -----------------------------------------------
        REM Erase the system menu

        ON ERROR GOTO L99
        ms1$ = "Is entry correct ?"
        ms2$ = "Select another record ?"
        ms3$ = "Save current record ?"
        OPEN FORM "u_public"

L11:    REM Select record
        REQUEST "Select Author",""
                    20,a%,a$,50,author.public
        SELECT FIRST
        IF a% = 0 THEN GOTO L12
        WHILE author.public <> a$
              SELECT NEXT
        WEND
        FORM

L111:REM Update the existing record
        MOUSE ON
        ENTER
        MOUSE OFF

L112:REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L111
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN
              STORE
              GOTO L11
        ELSE
              MOUSE ON
              GOTO L12
        END IF

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
```

```
        REQUEST ms3$,"",130,a%
        IF a% = 1 THEN STORE
        CHAIN "main"

L14:    REM Undo adding record
        SELECT CURRENT
        FORM
        GOTO L111

L99:    REM Error condition
        REQUEST ERR$ ( ERRNO )," Press OK to make another
              selesction",1,a%
        IF a% = 1 THEN GOTO L11
        GOTO L12
        END


        REM ------------------------------------------
        REM         I_MAIN
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author  : Partoyo
        REM         Created : September 8, 1990
        REM         Modified :
        REM ------------------------------------------
        REM Erase the system menu and display the opening screen
        MENU CLEAR
        OPEN FORM "i_main"

L1:     REM wait here for a pushbutton to be clicked
        FORM
        WAIT MOUSE
        GOTO L1

LIF:    REM Update Faculty
        CHAIN "i_fac"

LIC:    REM Update Course File
        CHAIN "i_course"

LID:    REM Update Department
```

```
        CHAIN "i_dept"

LIO:    REM Update Offering
        CHAIN "i_offer"

LIP:    REM Update Publication
        CHAIN "i_public"

EXT:    REM back to main program
        CHAIN "main"


        REM ------------------------------------------
        REM         I_FAC
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author   : Partoyo
        REM         Created  : September 8, 1990
        REM         Modified :
        REM ------------------------------------------
        REM Erase the system menu and display the blank form
        ON ERROR GOTO L99
        upd$ = "n"
        ms1$ = "Is entry correct ?"
        ms2$ = "Continue with data entry ?"
        ms3$ = "The new record has been deleted"
        MENU CLEAR
        OPEN FORM "i_fac"

L1:     REM Set up data entry
        BLANK FORM

L11:    REM Enter New record
        ON ERROR GOTO L99
        upd$ = "n"
        MOUSE ON
        ENTER
        MOUSE OFF
        REM ***** Verify wether the entry is correct or not
        REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L11
```

```
        GOTO L15

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        CHAIN "main"

L14:    REM Undo adding record
        IF upd$ = "n" THEN GOTO L1
        SELECT REMOVE
        REQUEST ms3$,ms2$,130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12

L15:    REM Insert record
        STORE
        upd$ = "y"
        REM **** Continue with data entry ?
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12

L99:    REM Error condition
        IF ERRNO = 57 THEN
                REQUEST "Record already exist","",2,a%
                RESUME L11
        ELSE
                REQUEST ERR$ ( ERRNO ), "Press OK to make another
                selection",1,a%
                CHAIN "main"
        END IF
        END


        REM ------------------------------------------
        REM             I_COURSE
        REM             Dept of Computer Science
        REM             Departemental Database
```

```
        REM
        REM             Author   : Partoyo
        REM             Created  : September 8, 1990
        REM             Modified :
        REM -----------------------------------------------
        REM Erase the system menu and display the blank form
        ON ERROR GOTO L99
        upd$ = "n"
        ms1$ = "Is entry correct ?"
        ms2$ = "Continue with data entry ?"
        ms3$ = "The new record has been deleted"
        MENU CLEAR
        OPEN FORM "i_course"

L1:     REM Set up data entry
        BLANK FORM

L11:    REM Enter New record
        ON ERROR GOTO L99
        upd$ = "n"
        MOUSE ON
        ENTER
        MOUSE OFF
        REM ***** Verify wether the entry is correct or not
        REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L11
        GOTO L15

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        CHAIN "main"

L14:    REM Undo adding record
        IF upd$ = "n" THEN GOTO L1
        SELECT REMOVE
        REQUEST ms3$,ms2$,130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12
```

```
L15:  REM Insert record
      STORE
      upd$ = "y"
      REM **** Continue with data entry ?
      REQUEST ms2$,"",130,a%
      IF a% = 1 THEN GOTO L1
      MOUSE ON
      GOTO L12

L99:  REM Error condition
      IF ERRNO = 57 THEN
            REQUEST "Record already exist","",2,a%
            RESUME L11
      ELSE
            REQUEST ERR$ ( ERRNO ), "Press OK to make another
            selection",1,a%
            CHAIN "main"
      END IF
      END

      REM ------------------------------------------------
      REM         I_OFFER
      REM         Dept of Computer Science
      REM         Departemental Database
      REM
      REM         Author   : Partoyo
      REM         Created  : September 8, 1990
      REM         Modified :
      REM ------------------------------------------------
      REM Erase the system menu and display the blank form
      ON ERROR GOTO L99
      upd$ = "n"
      ms1$ = "Is entry correct ?"
      ms2$ = "Continue with data entry ?"
      ms3$ = "The new record has been deleted"
      MENU CLEAR
      OPEN FORM "i_offer"

L1:   REM Set up data entry
      BLANK FORM

L11:  REM Enter New record
```

```
        ON ERROR GOTO L99
        upd$ = "n"
        MOUSE ON
        ENTER
        MOUSE OFF
        REM ***** Verify wether the entry is correct or not
        REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L11
        GOTO L15

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        CHAIN "main"

L14:    REM Undo adding record
        IF upd$ = "n" THEN GOTO L1
        SELECT REMOVE
        REQUEST ms3$,ms2$,130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12

L15:    REM Insert record
        STORE
        upd$ = "y"
        REM **** Continue with data entry ?
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12

L99:    REM Error condition
        IF ERRNO = 57 THEN
                REQUEST "Record already exist","",2,a%
                RESUME L11
        ELSE
                REQUEST ERR$ ( ERRNO ), "Press OK to make another
                selection",1,a%
                CHAIN "main"
```

```
        END IF
        END


        REM ------------------------------------------
        REM         I_DEPT
        REM         Dept of Computer Science
        REM         Departemental Database
        REM
        REM         Author   : Partoyo
        REM         Created  : September 8, 1990
        REM         Modified :
        REM ------------------------------------------
        REM Erase the system menu and display the blank form
        ON ERROR GOTO L99
        upd$ = "n"
        ms1$ = "Is entry correct ?"
        ms2$ = "Continue with data entry ?"
        ms3$ = "The new record has been deleted"
        MENU CLEAR
        OPEN FORM "i_dept"

L1:     REM Set up data entry
        BLANK FORM

L11:    REM Enter New record
        ON ERROR GOTO L99
        upd$ = "n"
        MOUSE ON
        ENTER
        MOUSE OFF
        REM ***** Verify wether the entry is correct or not
        REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L11
        GOTO L15

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        CHAIN "main"
```

```
L14:  REM Undo adding record
      IF upd$ = "n" THEN GOTO L1
      SELECT REMOVE
      REQUEST ms3$,ms2$,130,a%
      IF a% = 1 THEN GOTO L1
      MOUSE ON
      GOTO L12

L15:  REM Insert record
      STORE
      upd$ = "y"
      REM **** Continue with data entry ?
      REQUEST ms2$,"",130,a%
      IF a% = 1 THEN GOTO L1
      MOUSE ON
      GOTO L12

L99:  REM Error condition
      IF ERRNO = 57 THEN
            REQUEST "Record already exist","",2,a%
            RESUME L11
      ELSE
            REQUEST ERR$ ( ERRNO ), "Press OK to make another
            selection",1,a%
            CHAIN "main"
      END IF
      END


      REM ------------------------------------------
      REM         I_PUBLIC
      REM         Dept of Computer Science
      REM         Departemental Database
      REM
      REM         Author   : Partoyo
      REM         Created  : September 8, 1990
      REM         Modified :
      REM ------------------------------------------
      REM Erase the system menu and display the blank form
      ON ERROR GOTO L99
      upd$ = "n"
      ms1$ = "Is entry correct ?"
```

```
        ms2$ = "Continue with data entry ?"
        ms3$ = "The new record has been deleted"
        MENU CLEAR
        OPEN FORM "i_public"

L1:     REM Set up data entry
        BLANK FORM

L11:    REM Enter New record
        ON ERROR GOTO L99
        upd$ = "n"
        MOUSE ON
        ENTER
        MOUSE OFF
        REM ***** Verify wether the entry is correct or not
        REQUEST ms1$,"",130,a%
        IF a% = 0 THEN GOTO L11
        GOTO L15

L12:    REM Wait here for a pushbutton to be clicked
        WAIT MOUSE
        GOTO L12

L13:    REM Back to the main program
        CHAIN "main"

L14:    REM Undo adding record
        IF upd$ = "n" THEN GOTO L1
        SELECT REMOVE
        REQUEST ms3$,ms2$,130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12

L15:    REM Insert record
        STORE
        upd$ = "y"
        REM **** Continue with data entry ?
        REQUEST ms2$,"",130,a%
        IF a% = 1 THEN GOTO L1
        MOUSE ON
        GOTO L12
```

```
L99:    REM Error condition
        IF ERRNO = 57 THEN
                REQUEST "Record already exist","",2,a%
                RESUME L11
        ELSE
                REQUEST ERR$ ( ERRNO ), "Press OK to make another
                selection",1,a%
                CHAIN "main"
        END IF
        END
```

LIST OF REFERENCES

1. Elmasri/Navathe, *Fundamentals of Database System*, The Benjamin/Cumming Publishing Co., 1989.

2. David M.Kroenke and Kathleen A.Dolan, *Database Processing*, Macmillan Publishing Co., 1988.

3. C.J.Date, *An Introduction to Database System Volume I*, Addison-Wesley Publishing Co., 1990.

4. Jonathan S.Wall, *Semantic shortcomings of Database Management Systems Based on A Relational Model*, Master Thesis, Naval Postgraduate School, Monterey, California, June 1988.

5. Naval Postgraduate School Technical Report NPS52-88-050, *Implementation of Visual Database Interface using an Object-Oriented Language*, by C. T. Wu and D.K.Hsiao, June 1988.

6. Michael Hyman, *Microsoft Windows Program Development*, Management Information Source Inc., 1988.

7. Microsoft Corporation, *Microsoft User's Guide*, Microsoft Corporation, 1987.

8. Precision Software Limited, *Superbase-4 Database and Text Editor*, Precision Inc, 1990.

9. Ben Shneiderman, *Designing the User Interface Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishing Company, 1987.

10. Nan C.Shu, Harry K.T. Wong, Vincent Y.Lum, *Forms Approach to Requirements Specification for Database Design*, ACM, 1983.

11. Nan C.Shu, *Visual Programming*, Van Nonstrand Reinhold Company Inc., 1988.

12. Yao, S. B.,Hevner, A.R., Shi, Z., and Luo, D., *FORMANAGER: An Office Form Management System*, ACM Transaction of Office Information System, Vol.2, No.3 (July 1984), pp.235-262.

13. D.Tsichritzis, *FORM MANAGEMENT*, Communications of ACM, Vol.25, No.7, July 1982.

14. Judith R.Brown, Steve Cunningham, *Programming The User Interface*, John Wiley & Sons Inc., 1989.

15. Lum, V.Y., Choy, D.M., and Shu, N.C., *OPAS: An Office Automation System*, IBM Syst.J. Vol.21, No.3, 1982.

16. Alan Simpson, *Understanding dBASE IV*, SYBEX Inc., 1989.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 37 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor Thomas Wu, Code CS/Wq Naval Postgraduate School Monterey, California 93943-5000	1
5.	R.Griffin, Code GS/Gr Naval Postgraduate School Monterey, California 93943-5000	1
6.	Office of Defense Attache Embassy of the Republic of Indonesia 2020 Massachusetts Avenue, N.W. Washington, D.C., 2003	1
7.	Chief U.S. Defense Liason Group, Indonesia ATTN. Dispullahta TNI-AD Kol. Suharminto OMADP Box 2, APO San Fransisco 96356	1
8.	Ka Dispullahta TNI-AD Jl.Veteran No.5 Jakarta Pusat, Indonesia	1

9.	Asisten Operasi KASAD Jl. Veteran No.5 Jakarta Pusat, Indonesia	1
10.	Asisten Personil KASAD Jl. Veteran No.5 Jakarta Pusat, Indonesia	1
11.	Direktur Perhubungan Jl.S.Parman Jakarta Barat, Indonesia	1
12.	Akademi Manajemen Informatika dan Komputer (AMIK) Bunda Mulia Jl. AM Sangaji No.20 Jakarta Pusat, Indonesia	1
13.	Partoyo Jl. Pendidikan I/K8 Cijantung, Jakarta Timur, Indonesia	1